Studies in Particle Swarm Optimization Technique for Global Optimization

by

ARASOMWAN Akugbe Martins

Student No. 211560835

Submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy in Computer Science

at the

School of Mathematics, Statistics and Computer Science
University of KwaZulu-Natal
Durban, South Africa

December, 2013

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION

The research described in this thesis was performed at the University of KwaZulu-Natal under the supervision of Dr. A. O. Adewumi. I hereby declare that all materials incorporated in this thesis are my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed:

Arasomwan Akugbe Martins

Date: December 2013

As the candidate's supervisor, I have approved the Thesis for submission

Signed:

Dr. A. O. Adewumi

Date: December 2013

UNIVERSITY OF KWAZULU-NATAL

COLLEGE OF AGRICULTURE, ENGINEERING AND SCIENCE

DECLARATION - PLAGIARISM

I, _____declare that

1. The research reported in this thesis, except where otherwise indicated, is my original research.
2. This thesis has not been submitted for any degree or examination at any other University.
3. This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This thesis does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a. Their words have been re-written but the general information attributed to them has been referenced
    b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signed:

_____

Arasomwan Akugbe Martins

Dedication

This Thesis is dedicated to my darling wife, Mrs. Deborah A. Arasomwan,

Finally, it is dedicated to the Most High God, my Source, Salvation, Success, Sustainer and Security.

Acknowledgement

Firstly, I would like to appreciate my darling wife, Mrs. Arasomwan, A. Deborah for her prayers, patience, support and understanding all through the period of my studies, most especially because I have been far away from home.

My profound and sincere gratitude goes to my supervisor, Prof. A.O. Adewumi for making every effort to see to the successful completion of my Doctoral programme. Thank you for always doing your best and encouraging me to be my very best especially in mentoring me to a be a self-confident researcher. The exposure provided to optimization research and training on how to improve my research output is much appreciated.

Many thanks to Prof. M. M. Ali of the School of Computational and Applied Mathematics, University of Witwatersrand, Johannesburg, meeting you was a great privilege and added to my research potential in diverse ways.

My sincere appreciation also goes to the entire Optimization and Modelling Group, School of Mathematics, Statistics and Computer Science at UKZN, it was wonderful having those seminars together as a team as this provided the needed motivation to improve the quality of research work being done.

I am deeply grateful to the College of Engineering, Agriculture and Science, University of KwaZulu-Natal for being awarded research grants and for the conducive environment available for research. My gratitude goes to the management of the Federal College of Education, Okene Nigeria for granting me the permission to embark on my PhD study.

I would also like to thank Bro. Ikharo A. Briamoh and Bro. Olusanya, O. Micheal for the various contributions they have made to my Doctoral programme. To all brethren in the faith both in Okene, Kogi State, Nigeria and KwaZulu-Natal Province, South

Africa: Pastor and Mrs. A. O. Adewumi, Pastor Adesina Joseph, Pastor and Mrs. Omeiza, to mention a few, I say thank you for your prayers.

In addition, I would like to express my gratitude to all those who have in one way or the other contributed to this research work. To my dear sisters and brother, Adesuwa, Osareniye, Iziegbe, Uyinmwen and to your respective families; may God bless and keep you all.

Lastly, I am highly indebted to God Almighty for the ideas, help, mercy, preservation and favour which He has provided and that I have enjoyed throughout the course of this research.

Arasomwan A. Martins
December, 2013.

Abstract

This thesis presents a series of studies conducted on particle swarm optimization (PSO) technique for global optimization inspired by the drawbacks identified in the technique with respect to premature convergence, weak local search ability and the desire to make the technique simpler and more effective, efficient and robust when handling optimization problems with many local optima. Generally, PSO is widely applied by individuals, enterprises and researchers to seek best possible solutions to varieties of problems amidst limited resources hence the need to better efficiency in its implementation. Many variants of PSO have been proposed to address its drawbacks with varying successes and failures stories. Many of these variants have introduced several other parameters, complexities and more computational efforts into the technique, yet its drawbacks are not sufficiently addressed. Besides, there exist some areas, like particle velocity limits and search space limits of the PSO technique that remain static throughout its lifetime of execution in many existing variants. Introducing dynamism could make the algorithm perform better and obtain better quality solutions to optimization problems. Besides, the pure greedy method of obtaining the swarm global best among the personal bests of all the particles in the swarm is a common attribute of very many PSO variants. These form part of the things addressed in the studies carried out in this thesis.

In this study, selected existing PSO variants were improved upon and additional variants were proposed which greatly improved the efficiency and performance of the PSO technique. These proposed variants include Swarm Success Rate Decreasing Inertia Weight PSO ($SSRDIW_{PSO}$) and Swarm Success Rate Random Inertia Weight PSO ($SSRRIW_{PSO}$) which use swarm success rate as the feedback parameter for their inertia weight strategies to enhance the explorative and exploitative power of the PSO technique. The Modified Basic PSO (*M*-BPSO) with seven versions, which use dynamic velocity limits to control the global and local search activities of the PSO; Improved Original PSO (IOPSO) which does not use the inertia weight parameter but has dynamic search space and velocity limits; PSO_{CLUS} in which the PSO technique was hybridized with a novel local search technique called Collective Local Unimodal search (CLUS) and GRA-PSO which diversified the operations of the PSO by

incorporating randomness and adaptivity to complement the greedy method PSO, chooses the global best from among the personal bests of particles in the swarm.

Through numerical experiments, several test problems from literature were used to validate the proposed variants. The results obtained were compared with their original counterparts and with various efficient PSO variants that exist in literature. The results reveal that substantial evidence exist that prove that the new variants are better than their original counterparts and several of the PSO variants in literature in terms of reliability, robustness, convergence speed, solution quality, search ability and efficiency. As a result, the new variants proposed in this thesis offer an alternative to many currently available algorithms for solving global optimization problems in which the gradient information is not readily available. These variants can be applied to solve various global optimization problems and are available for optimization researchers. The results can also serve as a benchmark on which future researches could be based.

Table of Contents

List of Tables

List of Included Articles

1. **Article in Peer-reviewed Journal (ISI)**
   i. Arasomwan A. M. and Adewumi A.O. (2013), On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global optimization, *The Scientific World Journal*, Vol. 2013, 12 pages. DOI: http://dx.doi.org/10.1155/2013/860289
   ii. Arasomwan, M.A. and Adewumi, A.O. (2014). An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps. *Mathematical Problems in Engineering,* vol. 2014, Article ID 178959, 17 pages, 2014. doi:10.1155/2014/178959.
   iii. Arasomwan, M.A. and Adewumi, A.O. (2014). Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems. Special Issue on Bioinspired Computation and Its Applications in Operation Management (BIC), *The Scientific World Journal*, vol. 2014, Article ID 798129, 23 pages, 2014. doi:10.1155/2014/798129

2. **Articles under Review for Peer-reviewed Journals (ISI)**
   i. Arasomwan A. M. and Adewumi A. O. (2013), An Improved Particle Swarm Optimizer based on Swarm Success Rate for Global Optimization Problems. Submitted to *The Journal of Theoretical and Experimental Artificial Intelligence.*
   ii. Arasomwan A. M. and Adewumi A. O. (2013), Improved Particle Swarm Optimizer with Dynamically Adjusted Search Space and Velocity Limits for Global Optimization. Submitted to *Journal of Artificial Intelligence Tools.*
   iii. Arasomwan A. M. and Adewumi A. O. (2013), Improved Particle Swarm Optimization for Global Optimization with(out) some Control Parameters. Submitted to *International Journal of Bio-inspired Computing.*

3. **Articles in Peer-reviewed Conference Proceedings**
   i. Arasomwan A. M. and Adewumi A. O. (2013), On Adaptive Chaotic Inertia Weights in Particle Swarm Optimization, in *Proceedings of the IEEE*

Symposium on Swarm Intelligence (SIS), 2013, Singapore, April 2013, pp.72-79.

ii. Arasomwan A. M. and Adewumi A. O. (2013), An Adaptive Velocity Particle Swarm Optimization for High-Dimensional Function Optimization, in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, Mexico City, Mexico, June 2013, pp. 2352–2359.

Chapter 1

Introduction

Since its inception [35, 36], the Particle Swarm Optimization (PSO) technique has experienced tremendous improvements, which has fostered its wide application to optimization problems in different fields of study. The diverse trends accompanying researches in PSO include its hybridization with other optimizers, addressing the problem of premature convergence as well as the adaptation of its control parameters during optimization.

Control parameters like inertia weight, acceleration coefficients and random factors are widely acknowledged to play important roles in the performance of PSO and different mathematical analysis have been done relative to how these parameters influence the diversity of the particles, with the belief that population diversity influences optimization effectiveness. One of the utmost goals of the authors who introduced the PSO technique was to make it as simple as possible. Contrary to this, many existing PSO variants have introduced some computational complexities to this technique. However, there are some studies on simplified variants of PSO in literature. One of the objectives of this research work is to provide answers to the following questions:

i. Are there other ways, different from the existing methods, that the operations of the original PSO and the basic PSO can be altered without affecting the velocity and position updating formulas of particles for increased performance compared to the existing PSO variants?
ii. Can the exploration and exploitation activities of the PSO technique be successfully and efficiently achieved without using the widely accepted inertia weight parameter? In reality, is it possible to implement the PSO technique without any of the three control parameters (inertia weight, acceleration coefficients and random factors) in the velocity updating formula without

additional complex computational efforts being exerted elsewhere in the algorithm?

iii. Can the existing simplified PSO variants be made simpler without compromising the efficiency, accuracy, reliability and robustness needed in the discovery of the global minimum compared with existing variants? In this context, efficiency refers to the amount of efforts (CPU time or number of function evaluations) required to obtain a solution. Accuracy describes how close the final solution obtained by a global optimization algorithm is to the known global minimum of a problem while reliability explains how successful the method is in finding the global minimum.

This thesis reports series of studies carried out on basic PSO and different existing PSO variants to improve on their identified weaknesses in order to target better global optimal results. As a result, new hybrids and promising variants of the PSO technique, including derived simplifications, were proposed which greatly improved on the efficiency of existing PSO techniques. These variants can be applied to solve various global optimization problems and are available for optimization researchers. Several problems were used to validate the proposed variants and their results were compared with various efficient PSO variants that exist in literature.

1.1 Background to the Study

Inherent in the human nature is the quest for the best possible in almost all endeavours of life. Driven by this kind of nature, individuals, enterprises and governments daily seek optimal solutions, amidst limited resources, to different problems encountered. Many of these problems can be formulated as optimization problems, thus optimization plays an increasingly significant role in daily management and technical decision making. Examples in science would include varying some decision parameters to maximize profit (e.g. investment portfolios, supply chains, etc.); in engineering, choosing design parameters to improve some objective and in data analysis, extracting some model parameters from data while minimizing error measures (e.g. fitting).

Global optimization is an inherently difficult problem because no general criterion exists for determining when the global optimum is reached. This type of optimization

seeks to provide solutions to optimization problems which are often multi-modal and non-convex. These solutions may be good globally or may be a mix of globally and locally good solutions. Generally, optimization problems entail how to select the best course of action given some restrictions.

A large number of optimization methods for solving various optimization problems exist in literature. Broadly, these methods can be categorized into either local or global optimization search techniques. A local search method iteratively improves its estimate of the optimum by searching for better solutions within the local neighborhood of the current solution while a global search method searches complicated landscapes of multiple local minima. Nature always finds the optimal solution to solve its problem maintaining perfect balance among its components and is thus equipping man to discover various inspired solutions for problem-solving and adaption to the ever-changing environment. For researchers, inspirations from natural systems that display problem-solving capabilities have been received to develop algorithms for solving complex and challenging optimization problems. Nature-inspired techniques have been evolving in recent years, exhibiting extremely diverse, dynamic, robust, complex and fascinating phenomenon.

Since the inspiration for each nature-inspired technique are unrelated to a particular class of optimization problems, it becomes easier to modify them substantially especially when applied in practice. With amazing results being obtained by researchers, the scope and viability of nature-inspired algorithms has been broadened opening up the possibilities for exploring new areas of application and providing more opportunities in computing. A major reason why these sets of algorithms have become popular is because they are easy to code in relatively few lines. They have become an important part of contemporary research in global optimization algorithms, computational intelligence and soft computing

Swarm intelligence (SI) is one of the classes of nature-inspired metaheuristics that has been used to provide (near) optimal solutions to many complex optimization problems in recent years. The goal of SI is to design intelligent multi-agent systems by taking inspiration from the collective behavior of social organisms. Amongst the first set of (and most popular) SI metaheuristics is the PSO. PSO [35,36] is a technique that

displays problem-solving capabilities that enables researchers solve complex and challenging optimization problems. It is an evolutionary computation technique inspired by the social behaviour of birds and schools of fish.

The basic idea of the PSO stems from biology where a swarm of birds are able to coordinate themselves, with some degree of randomness, in order to achieve a goal. Each particle (bird) uses the local information regarding the displacement of its reachable close neighbours to decide on its own displacement, resulting in complex and adaptive collective behaviours. The concept was introduced to the field of optimization in 1995 [35, 36]. PSO can be used to provide solutions to optimization problems with multimodal or unimodal landscapes.

When PSO was initially proposed, swarm size, particle velocity, acceleration coefficients and random coefficients, were the associated parameters that controlled its operations and efficiency. However, it exhibited the problem of premature convergence. In ridding the PSO of this problem and make it more efficient, many variants have been developed and these are detailed in literature [41, 55]. These variants have additional parameter(s) and require extra (complex) computational effort(s), which give them an edge over the Original OPSO (OPSO).

1.2 Motivation

Optimization problems are wide in range and numerous, hence methods required to solve them require active and dynamic researches. These include data mining, engineering, and bio-computing problems which are large-scale in terms of the decision variables that need to be handled in trying to solve them. As a result, the performance of most available optimization algorithms deteriorate very quickly as the the problem dimension increases. This is because complex problems has large solution space which increases exponentially with the problem size. A current trend is to develop scalable algorithms with efficient search strategies to explore all the promising regions in the solution space within given constraint. Nature-inspired optimization techniques are prominent among these algorithms. They are presently more frequently studied and utilized for solving optimization problems in academia and industry than mathematical optimization techniques such as convex

programming, linear programming and other metaheuristics [79, 123] due to the increased complexities of many real-world problems.

It is a general knowledge that nature-inspired metaheuristic algorithms are prominent in tackling challenging highly nonlinear optimization problems with evidence of efficiency. As a result, researches are expanding towards this direction in different fields. However, till date, researchers have only utilised very limited characteristics inspired by nature; thus, other properties of natural environments are worth investigating for the development of novel nature-inspired algorithms. Moreover, there are still lots of opportunities to improve existing nature-inspired techniques hence the thrust of this paper.

The nonlinearity of several optimization problems often results in multiple local optima that pose substantial challenges in obtaining the global optimality of interest. Therefore, the need for efficient techniques and improvement on existing ones to solve complex global optimization problems in the continuous space is evident. PSO being one of the popular techniques used to solve both simple and complex optimization problems has undergone countless modifications and improvements since when it was introduced; hence, many of its variants exist. These modifications and improvements are done either on the parameters that control the operations of PSO, the addition of new parameters or both, the resulting variants that have been useful in solving many global optimization problems.

Diverse variants of the PSO have been proposed with varying level of improved performances [93,133]. However, many of these variants are characterized by: static particle search space and velocity limits, which limit their flexibilities in obtaining optimal solutions for many of the optimization problems. Furthermore, in spite of some extra computations inherent in these variants and additional parameters like inertia weight incorporated, premature convergence, which is the major challenge associated with the OPSO technique, remains a problem that many of the variants have not been able to handle successfully [29, 39, 41, 46]. In cases where (near) optimal solutions are obtained, they are with low precision [53, 132].

In many of the PSO variants, solution search space and velocity threshold are static throughout the execution of the algorithm [4, 5, 37, 41, 77]. This characteristic somewhat limits the flexibility of these variants in obtaining optimal solutions for many of optimization problems. Also, there is the challenge of selection of velocity threshold especially when dealing with some practical problems. Trial-and-error approach which can be computationally intensive and time consuming may be required to make the selection. . There is need for more dynamic way of varying the solution space and velocity threshold in order to obtain optimal results with higher precision for optimization problems when using PSO and its variants. This can be done based on the state of the particles' dimensions so as to enable the algorithm concentrate its search on the sub-range defined during its execution instead of the entire search space all the time. In addition, this could also enable the algorithm escape premature convergence so as to obtain better quality solutions to given optimization problems.

The inertia weight parameter [96] was introduced into PSO to enable it obtain better results to optimization problems by balancing the algorithm's exploration and exploitation activities. Many Inertia Weight Strategies (IWS) have the initial and final values of the inertia weight fixed, thereby ruling out the flexibility of obtaining lower or higher values for the inertia weight that could help the algorithm obtain good optimal results. Also, many of the IWS do not have access to information about the state of the swarm in the solution search space; this could influence the nature of the search for optimal solutions by the swarm. Therefore, it is of utmost importance that a means of realising the state of the swarm in the search space is devised, in addition to creating some flexibility in either of the limits of the IWS with the belief that this could help the algorithm obtain better results.

To further enhance the performance of the PSO algorithm in this work, randomness was introduced into its IWS. Since chaotic activities can play the role of randomization, this has been brought into the IWS with the logistic chaotic map being more prominently used [69]. This feature has improved the optimizing capability of PSO by introducing better global search mobility. However, there are several chaotic maps in literature that have the possibility of enhancing the performance of PSO even more than the logistic map. Therefore, the effects of other chaotic maps on the

performance of PSO algorithms need further investigation. The outcomes obtained could provide some useful information to optimization practitioners in choosing chaotic maps to apply in the various IWS hence another focus of this work

1.3 Aim and Objectives

The primary aim of this thesis is to simplify the basic PSO technique and enhance selected existing PSO variants so as to improve their performances and extend their scope of applicability to optimization problems. The objectives summarized below provide guidance in achieving this aim.

i. To study the parameters of the PSO technique to better understand their individual contributions to the algorithm's operations and efficiency and to device means of adjusting these parameters to further enhance the efficiency of the PSO. Some of these will be achieved by introducing dynamism into some static aspect of the current PSO algorithm and implementations.
ii. To develop PSO variants with enhanced parameter selection and combination techniques.
iii. To introduce adaptivity and randomness into the method of selecting the swarm global best from among the personal bests of particles, instead of the commonly used greedy method.
iv. To develop PSO variants that could efficiently handle high dimensional global optimization problems.
v. To develop an improved PSO hybrid with local search that compete significantly well with current variants.

1.4 Scope of the Thesis

This thesis considered the PSO technique for solving both simple and complex optimization problems in the continuous space. Much attention was given to the parameters of the technique because of the vital roles they play in the operations of the technique. The target is to develop means of making the PSO technique simpler and more efficient than existing variants in handling optimization problems. Conclusions and remarks are based on extensive simulation studies of the proposed variants which are compared with that in literature. The set of benchmark problems

used in all the studies are with diverse characteristics and complexities as found in literature and real world problems.

1.5 Methodology

In this work, different variants of the PSO algorithms were developed. Numerical simulations were carried out on these variants using various benchmark test problems. Empirical results obtained from the studies were analyzed using statistical techniques to demonstrate the superiority of the new variants in terms of their performances compared to the performance of existing PSO variants in literature.

1.6 Contributions

This research study carried out series of investigations on the different parameters and their contributing effect to the PSO technique for global optimization with the aim of addressing its major drawback and to improve its efficiency. Some of the parameters are inertia weight, particle velocity limits and acceleration coefficients, which play prominent roles in optimizing the power and efficiency of the PSO in the course of obtaining (near) optimal solutions for global optimization problems.

During the research process, some existing PSO variants were improved upon and new hybrids and promising variants of the PSO were proposed which greatly improved the efficiency of the PSO technique. Some of the research contributions in the work are highlighted below:

1. The exploratory and exploitative powers of selected existing PSO variants were improved upon by introducing the swarm success rate as the feedback parameter for their inertia weight strategies.
2. Some PSO variants without the inertia weight parameter were proposed. These variants implemented dynamic velocity clamping of particles and dynamic solution search space.
3. A variant which diversified the operations of the PSO by incorporating randomness and adaptivity to complement the greedy way PSO chooses the personal and global best of particles was proposed.

Several problems were used to validate the proposed variants and their results were compared with various efficient PSO variants that exist in literature. From the results obtained from this research study, there are indication of significant success with proposed variant which we hope would be useful for both researchers and practitioners in the field of global optimization. These variants can be applied to solve various global optimization problems and are available for optimization researchers. Finally, results obtained in this study further provided higher benchmarks on which further work on PSO can be based.

1.7 Thesis Outline

The rest of the thesis is organized as follows:

Chapter 2 provides an introduction to the theory of optimization and metaheuristics, followed by a review of the existing swarm intelligence (SI) techniques with emphasis on the PSO technique.

Chapter 3 presents the PSO based on dynamic velocity and search space limits. Several experiments were conducted using these limits to improve the performance of the PSO. Two papers developed from the results of the experiments are also presented.

Chapter 4 examines the effect of the swarm success rate feedback parameter and the chaotic maps on the performance of the PSO algorithms. Numerical simulations were performed to obtain results used for analyzing the effect of the feedback parameter as well as the chaotic maps. The results obtained are compared with those of selected existing PSO variants. To conclude this chapter, three papers which are products of the research study are included.

Chapter 5 presents certain simplifications of the PSO technique. These are done without compromising the performance of the PSO. Experimental results and comparisons with existing PSO variants are presented. One research paper developed is included in this chapter.

In chapter 6, a new local search technique was proposed and used to improve the PSO algorithm. The technique was applied to existing PSO variants and results obtained were compared with those of the prevailing variants to verify the suitability of applying the local search. In addition, a different method for updating the positions of particles was devised and implemented. One research paper is included in this chapter.

Chapter 7 presents a simple PSO variant that is able to handle high dimensional problems. The variant adaptively adjusts the velocity of particles based on Euclidean distance between the position of each particle and the position of the global best particle and applied to continuous optimization problems with low (10 – 30) and high (50 – 4,000) dimensions.

Finally, Chapter 8 presents a summary and conclusion of this thesis. Major contributions are highlighted in addition to suggestions for future research.

Chapter 2

Literature Review

2.1. Optimization

Optimization encompasses selecting the best course of action(s) among several alternative while considering given restrictions. Examples of practical optimization problems includes: production of fuel efficient car, selecting portfolio that minimizes risks while maximizing returns, deciding the shortest route among several alternatives, to mention a few. These problems typically have three fundamental components namely, the objective function, decision variables, and constraints. The objective function is the numerical quantity to be optimized (maximized or minimized), that is, for which we seek the best possible value. Example may include maximizing expected return on a stock portfolio, minimizing production cost of an item, minimizing cost of travel through a given number of cities, and so on. The decision variables are quantities whose values can be manipulated in order to fulfill the objective function. Examples include quantities of stock to purchase and production schedule to optimize. Finally, constraints are simply restrictions that are placed on the possible values that decision variables can take. For instance, individual cannot invest more than s/he has into stock, a lecturer cannot teach more than one modules simultaneously. Within this broad framework, the complexity of optimization problems depends on many other factors that characterize the problems including type of decision variable and the nature of the objectives functions and/or constraints.

Generally, an optimization problem can be represented as follows:

$$\text{Optimize } f(\vec{x})$$

subject to

$$g_j(\vec{x}) \leq 0$$

where $\vec{x} = (x_1, x_2, \dots, x_n)$ is the decision variable in $\mathfrak{R}^n$, $f: \mathfrak{R}^n \to \mathfrak{R}$ is the objective function and $g_j: g_j: \mathfrak{R}^n \to \mathfrak{R}$, are the constraint functions, $j = 1, 2, \dots, m$.

The goal is to maximize or minimize the objective function. Solution $\vec{x}^*$ is a global minimizer of $f(\vec{x})$ if and only if $f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x}$ in the domain of $f(\vec{x})$. It is a global maximizer of $f(\vec{x})$ if and only if $f(\vec{x}^*) \geq f(\vec{x})$ for all $\vec{x}^*$ in the domain of $f(\vec{x})$. Optimization problems are often multi-modal due to having multiple good solutions, which could all be globally good or a mix of globally good and locally good solutions.

Researchers are often faced with numerous nonlinear multimodal optimization problems such as parameter optimization. Global optimization seeks to find the globally best solution for nonlinear models among multiple local optima. To formulate global optimization problems, it is assumed that, (i) objective function and constraints are continuous functions, (ii) the component-wise bounds related to the decision variable vector are finite, and (iii) the feasible solution set is not empty. Such problems require global search technique to solve them. The different approaches to solving global optimization problems can be broadly grouped into exact and heuristics methods. However, exact methods often fail to obtain the global optima in the face of complex optimization problems while the heuristics seek to compare accuracy slightly for speed in order to obtain a near-optima solution to such problems. An example of heuristic methods is the evolutionary algorithms which mimic the principle of biological evolution like natural selection and the "survival of the fittest". The different types of evolutionary search methods are made up of approaches that are aimed at continuous global optimization problems, and others that are targeted towards solving combinatorial problems.

Local optimization, on the other hand, involves finding an optimum solution within a neighbourhood set of candidate solutions as against all set of possible solutions. This involves the use of local search methods which apply a local perturbation within the neighbourhood in search of the optimum solution. These techniques are widely used to provide solutions to many NP-hard problems in various fields and are also useful in improving the search trajectories of global search techniques for better results, in most cases as hybrids. We explore this hybrid approach for PSO in this work. Examples of local search techniques include hill climbing, tabu search, simulated annealing, 2-opt algorithm for traveling salesman problem and the

collective local unimodal search (CLUS) which is explored in this work. Local optimization algorithms generally depend on the derivatives of the cost function and constraints to aid in the search. It also depends on an initial point which determines the result obtained.

Generally, there are many optimization algorithms which can be classified based on various factors and depending on the focus and characteristics of such algorithms. A typical classification [121] is presented in Table 2.1. A general term used for most heuristic-based search algorithms is metaheuristics among which is the focus of this work. We shall give a brief overview of these in the next session.

Table 2.1: Classification of optimization algorithms [121]

S/N	Focus/Characteristics	Class of Algorithms
1	Derivative or gradient of a function	i. Gradient-based ii. Derivative-free
2	Number of agents	i. Trajectory-based ii. Population-based
3	Search procedure/Movement	i. Deterministic ii. Stochastic
4	Search capability/space	i. Local ii. Global

2.2. Metaheuristics

Metaheuristics, a term coined by Glover in 1986 [49], are higher level black boxes designed to select, generate or find lower level heuristics that can provide good solutions to optimization problems. They do not guarantee that a global optimum point will be found but seek a reasonable trade-off between solution quality and computing time. Metaheuristics seek to maintain an intelligent balance between exploration and exploitation capabilities of the underlying heuristics while navigating the search space in search of near-optimal solutions.

Metaheuristic techniques are well-known global optimization methods which attempt to mimic some characteristics of natural phenomena or social behaviour and sometimes incorporate complex learning processes. The algorithms are approximate and usually non-deterministic. Several of these non-problem specific techniques have been proposed for global optimization and have helped to increase the overall computational efficiency for some large-scale problems [79]. Generally, most research on metaheuristics are based on empirical studies (as in this work) with a few exploring formal theoretical issues such as convergence.

2.2.1 Characteristics and Classification of Metaheuristics

As mentioned earlier, metaheuristics have many characteristics features [121]. As they guide and/or modify other lower level heuristics to produce solutions, metaheuristic algorithms often use some tradeoff between randomization and local search. While they are good in finding near-optimal solution within reasonable time, they do not guarantee that optimal solutions can be reached. Often, metaheuristics incorporate strategies to assist underlying heuristics escape from local optimum through proper combination of intensification (exploitation) and diversification (exploration) [121]. The latter helps the algorithm to explore the entire global search space in search of optimal solution while the former helps to concentrate on a local region of the search space in search of solution better than the current local optimum. A good metaheuristic aims to seek a good balance between intensification and diversification during search in order to improve convergence of the algorithm. This is to ensure that global optimality is achievable [121].

Metaheuristic algorithms including SI algorithms are optimization methods designed in accordance with the strategies laid out in the metaheuristic framework. They form essential part of contemporary global optimization algorithms and have been shown to be efficient with many advantages over traditional, deterministic methods [120]. These algorithms can be categorized as a constructive approach or a local search method [100]. A constructive algorithm builds solutions from scratch by gradually adding solutions' components to the initially empty solutions whereas on the other hand, local search algorithms start from a known solution and try to improve on it over time. Similarly, metaheuristic algorithms can be classified as single solution

based or population based depending on the number of solutions the algorithm act upon at each iteration. Evolutionary algorithms are population-based algorithms.

Metaheuristic frameworks are usually defined in general terms without dependence on problem specific characteristics such as requiring constraints or objective functions to be defined in certain form. This makes them fit into most real-life optimization problems with varying requirements, constraints or formulation. These features make metaheuristic algorithms more flexible compared to exact methods but they have to be adapted to problem-specific domain sometimes to achieve good performance [120].

2.3. Swarm Intelligence (SI)

This is a class of nature-inspired algorithms with potency for handling complex optimization problems. These algorithms currently have great impact in contemporary computing and this will continue even for future generation computing. A swarm is a collection of large number of homogenous, simple agents that interact locally with and their environment with no central control. SI is a research field that studies the emergent collective intelligence of self-organized and decentralized simple agents. The inspiration often comes from the social behaviours that are observed in nature, especially in social animals such as flocks of birds, fish schools and swarm bees. The social interactions among swarm individuals can either be direct or indirect. Examples of direct interactions are through visual or audio contacts, such as the waggle dance of honey bees. Indirect interactions occur when one individual changes the environment for others to respond to, for example, the pheromone trails of ants which they deposit as they search for food sources [48].

SI algorithms seek to mimic the natural or artificial collective behaviour of decentralized, self-organized systems. They are population-based techniques based on agents that interacts locally and with their environment. This local behaviour with some level of randomness of interacting agents often leads to an intelligent emerging behaviour that tends towards a global optimum. A concise introductory overview of the successes of some nature-inspired metaheuristics can be found in [125].

2.3.1 Properties of Swarm Intelligence Paradigm

Researchers have long realized the importance of emergent behaviour for complex problem solving especially in search of intelligent solutions to real-world problems. However, some recent advances in SI, comprising new swarm-based optimization methods, hybrid algorithms and innovative applications can be found in [82]. The major concepts underlying the SI research field are decentralization, stigmergy, self-organization, emergence, feedbacks (positive and negative), fluctuations and bifurcations. Furthermore, division of labor, morphogenesis and collective decisions are essential concept to the SI paradigm [48]. SI algorithms are population-based but not population-based techniques are swarm-based [120].

A typical SI system has the following properties [33,120]:

i. It is based on population of individuals which are relatively homogeneous (i.e. they are either all identical or they belong to a few typologies).
ii. Individuals interact based on simple behavioral rules that exploit local information they exchange directly or via the environment (stigmergy).
iii. Information exchange is through models of well-known behavior of the underlying agents such as chemical secretion, dance, or broadcasting ability depending on the nature of the agents.
iv. The overall emerging (global) behaviour of the system results from the self-organizing ability through the local interaction with each other and the environment.
v. There is no central control among the self-interested agents.

2.3.2 Swarm Intelligence Models

SI models are computational models inspired by natural swarm systems. Many SI models have been proposed and successfully applied in literature based on the characteristics of different natural swarm systems [33]. These include Altruism algorithm [115], Ant Colony Optimization [34], Artificial Immune System [33, 65, 112], Artificial Bee Colony [60], Bacterial Foraging [32, 83], Bat Echolocation [126], Cat Swarm Optimization [26], Charged System Search [61, 62, 109], Cuckoo Search [119], Firefly Algorithm [123, 124], Glowworm Swarm Optimization [66], Intelligent

Water Drops [78, 80, 98, 99], Mosquito Host-seeking [42], Particle Swarm Optimization [36, 63], River Formation Dynamics [87-89], Roach Infestation optimization [52], Slime Mould Optimization [75] and Stochastic Diffusion Search [17, 18, 74].

Table 2.2 provides, in alphabetical order, a list of well-known SI algorithms as well as motivation that lead to their derivations including their originators.

Table 2.2: List of Swarm Intelligence Algorithms and their motivations

S/N	ALGORITHM	MOTIVATION	RESEARCHER(S)	YEAR
1	Altruism Algorithm [115]	Hamilton's rule of kin selection	Waibe M., Floreans D & Keller L.	2011
2	Ant Colony Optimization [34]	The foraging behaviour of social ants	Dorigo, M	1992
3	Artificial Immune Systems [65]	The characteristics of the immune system of mammals	Kephart J. O.	1994
4	Artificial Bee Colony [60]	The foraging behaviour of bees	Karaboga, D.	2005
5	Bacterial Foraging [83]	The social foraging behaviour of bacteria such as *Escherichia coli*	Passino, K. M.	2002
6	Bat Echolocation [126]	Based on the echolocation behaviour of bats	Yang, X.-S.	2010
7	Cat Swarm Optimization [26]	Based on two of the major behavioral traits of cats termed ”seeking mode” and ”tracing mode”	Chu, S.-C. & Tsai, P.-W.	2006
8	Charged System Search [61]	Some principles from physics (laws of Coulomb and Gauss from electrostatics) and mechanics (Newtonian laws)	Kaveh, A. & Talatahari, S.	2010
9	Cuckoo Search [119]	The brooding behaviour of some cuckoo species, which use host birds to hatch their eggs and raise their chicks	Yang , X.-S. & Deb, S	2009
10	Firefly Algorithm [123]	The flashing patterns and behaviour of fireflies.	Yang, X.-S.	2008
11	Glowworm Swarm Optimization [66]	The behaviour of glowworms	Krishnanand, K. N. & Ghose, D.	2006

12	Intelligent Water Drops [98]	Natural rivers and how they find almost optimal paths to their destination.	Shah-Hosseini, H.	2007
13	Mosquito Host-seeking [42]	The host-seeking behaviour of mosquitoes	Feng, X., Lau, F. C. M. & Yu, H.	2013
14	River Formation Dynamics [87]	The way water forms rivers by eroding the ground and depositing sediments; similar to ant colony optimization.	Rabanal, P., Rodriguez, I, & Rubio, F.	2007
15	Roach Infestation Optimization [52]	Social behaviour of cockroaches	Havens, T. C. Spain, C. J., Salmon, N. G. & Keller, J. M.	2008
16	Particle Swarm Optimization [63]	Social behaviour of flock of bird and school of fishes	Kennedy, J. and Eberhart, R.. C.	1995
17	Slime Mould Optimization [75]	The lifecycle of amoeba	Monismith, D. R. & Mayfield, B. E.	2008
18	Stochastic Diffusion Search [15]	The restaurant game	Bishop, J. M.	1989

The swarm-based algorithms can be classified into three: microscopic agents-based SI, inanimate agents-based SI metaheuristics and others. Those based on microscopic agents are the Artificial Immune System, Bacterial Foraging, Slime-Mould while those based on inanimate agents are Charged System Search, River Formation Dynamics and Stochastic Diffusion Search. They are so named because they are unlike other SI metaheuristics like Ant colony Optimization, Artificial Bee Colony, Bat Echolocation, Cat Swam, Cuckoo Search, etc. stated in Table 2.2 which are based on animate agents individually visible to the human eyes.

2.4. Particle Swarm Optimization

PSO is one of the two fundamental mainstreams of SI developed in 1995 by James Kennedy and Russell Eberhart [36, 63]. It is a robust population-based stochastic optimization technique based on the social behavior, movement and intelligence of flocks of birds or schools of fish. It applies the concept of social interaction of a number of agents (particles) that constitute a swarm moving around, with a certain velocity, in an *n*-dimensional search space in search of the best solution to an optimization problem. Each particle resides at a position in the search space with the

fitness value (evaluated by the fitness function to be optimized) of each particle representing the quality of its position.

2.4.1 The original PSO framework

PSO involves a swarm of particles (agents) randomly initialized as points in the *n*-dimensional Euclidean space in search of optima solution to an optimization problem. Each particle *i* is characterized by a position vector $\vec{x}_i = (x_{i1}, \dots, x_{in})$, a velocity vector $\vec{v}_i = (v_{i1}, \dots, v_{in})$, and another position vector $\vec{p}_i = (p_{i1}, \dots, p_{in})$, which is the best position the particle has been able to find. The position of each particle is evaluated using the problem-specific objective function to determine their quality (fitness). As the particles move in the search space, their position and velocity vectors are updated as shown in equations (2.1) and (2.2) respectively in the OPSO algorithm.

$$\vec{v}_i^{t+1} = \vec{v}_i^t + c_1\vec{r}_1(\vec{p}_i - \vec{x}_i^t) + c_2\vec{r}_2\left(\vec{p}_g - \vec{x}_i^t\right) \quad (2.1)$$

$$\vec{x}_i^{t+1} = \vec{x}_i^t + \vec{v}_i^{t+1} \quad (2.2)$$

where $i = 1, 2, \dots, S$ and $t = 1, 2, \dots, T$; S represents the swarm size while T represents the maximum number of iteration allowed for the algorithm to run.

The velocity updating formula

Equation (2.1) known as the velocity updating formula is an integral part of the OPSO algorithm. This formula determines the flying speed of particles in the search space and is made up of the past velocity ($\vec{v}_i^t$), cognitive ($c_1\vec{r}_1(\vec{p}_i - \vec{x}_i^t)$) and social ($c_2\vec{r}_2\left(\vec{p}_g - \vec{x}_i^t\right)$) components. These three components play vital and different roles for PSO in demonstrating efficient optimizing power in providing (near) optimal solutions to various optimization problems, therefore making the algorithm sensitive to these parameters. The random coefficients $\vec{r}_1$ and $\vec{r}_2$ are *n*-dimensional vectors of uniform random numbers between 0 and 1, which introduce randomness to the searching strategy and enable the algorithm escape from local optima. The acceleration coefficients c_1 and c_2, also known as the cognitive and social scaling parameters respectively, determine the magnitude of the random forces on particles in the direction of $\vec{p}_i$ (the best position particle *i* has been able to find till the t^{th} iteration) and $\vec{p}_g$ (the overall best position the whole particles has been able to find till the t^{th} iteration). They play active roles in the convergence of the algorithm. The combined effort of these control parameters grants the velocity of each particle its value which

in turn determines the exploratory power of the algorithm. Getting appropriate values for particles' velocities demand additional computational efforts and time. An example can be found in [4].

It is very rare to find any PSO variants in literature that does not utilize the velocity updating formula, whether in its simplified form or otherwise. This confirms the implicit belief that PSO algorithms cannot be separated from the velocity updating formula for a successful optimization process.

The position updating formula

The position updating formula of each particle is made up of two components, namely: (i) previous position of each particle, (ii) current velocity of each particle. Depending on the value if its current velocity, each particle moves from its current position to another position in the solution space. The solution space is bounded by the upper and lower limits of the decision variables. During execution of the PSO technique, there is the possibility that values of the design variables extend beyond their lower ($X_{\min}$) and upper ($X_{\max}$) boundaries values which could lead to divergence. In such situations, the common practice is to artificially bring the affected particle back to the search space boundary as shown in Equation (2.3).

$$x_i = \begin{cases} X_{min}, & if\, x_i < X_{min} \\ X_{max}, & if\, x_i > X_{max} \end{cases} \qquad (2.3)$$

PSO technique has a wide range of applications in different fields including economics, engineering, industry, biology and many other complex real world optimization problems [3, 56, 73, 76].

2.4.2 Strengths and weaknesses of the PSO

The good attributes of the PSO technique has been a major attraction for numerous researchers. However, there are some challenges associated with its usage which have caused some alternative optimization techniques to be sought for in solving certain complex optimization problems. We present a few of the strengths and weaknesses of the OPSO in the following sub-sections.

2.4.2.1 Strengths of the PSO technique

PSO is self-adaptive, not problem-dependent and easy to implement with few parameters to adjust and/or optimize. It can be applied to solve both simple and complex optimization problems with less computational burden. As an intelligent technique, it does not need major adjustments to adapt it to new problems.

The technique does not need the gradient, continuity or differentiability of the problem to work with. It is also insensitive to problem dimensionality as well as initial solutions and can easily be parallelized for concurrent processing.

PSO has good global search ability with high accuracy and fast searching speed. Besides, it adopts real number representation which is decided directly by the solution.

2.4.2.2 Weaknesses of the PSO technique

In spite of the attractive features of the PSO as a potential global optimizer, some weaknesses associated with it have been identified by various researchers. These include:

i. *Lack of PSO variants that perform well in optimizing diverse set of problems*. As identified by [119], some variants of the PSO have high quality performance in solving complex multimodal functions but demonstrate unsatisfactory convergence rates in unimodal functions.
ii. *Victim of premature convergence (easily trapped in local optima) when solving complex multimodal problems*. This area of the PSO has received much attention in literature [4, 28, 43, 53, 54, 106, 131] and is a situation where the particles converge to the existing global best in the search space rather than the global optimum. This comes about because the more the particles communicate information to one another, the more similar they become especially when other particles follow are in line with the global best particle. A primary reason advanced by [106] that could also cause premature convergence is that, all particles have very similar behaviours because they have the same acceleration coefficients and inertia weight values leading to poor population diversity among the particles.

iii. *Profoundly dependence on the settings of cognitive and social learning constants as well as inertia weight* [53]. The cognitive and social learning constants (c_1 and c_2) were part of the OPSO [36, 63] while the inertia weight parameter became part of the technique in 1998 [104]. In [63], the stochastic factors of both the cognitive and social components were multiplied by c_1 and c_2 respectively and both constants were set to the value of 2.0, to give each factor a mean of 1.0. The inclusion of c_1 and c_2 in the PSO, their settings and contributions to accelerating convergence as well as enabling PSO to avoid local minimum [43, 63] has made these parameters fundamental to the operations of PSO. Also, the general belief that the inertia weight parameter is vital in balancing the exploration and exploitation activities of the PSO has equally made it an indispensable parameter.

iv. Possible computational inefficiency as measured by function evaluations [131].

v. *Blindness and computational inefficiency in the search process*. The cognitive and social components in the velocity update formula are weighted by c_1 and c_2 having values of 2.0 and r_1 and r_2 which take random values in the range [0,1]; this means that, these two weighting factors arbitrarily take values in the range [0,2]. This constrains the search covering the surrounding regions [0,2] to be centered on $\vec{p}_i^t$ and $\vec{p}_g^t$. Thus, while the search is approximating the global optimal solution, large weighting factors generated randomly could make the particles blindly jump over the optimal solution. On the other hand, small weighting factors generated randomly could result in an increase in the number of iterations needed to reach the global optimal solution especially if the search initially began far from the global optimal solution [63, 68]

vi. Slow convergence [7]

2.4.3 Developments and improvements on OPSO

The OPSO opened a new world of opportunity in the field of optimization. Over the years, many researchers in the field of optimization have made tremendous efforts to address the weaknesses of the PSO technique by developing different strategies to improve on its effectiveness, efficiency and robustness in handling optimization problems. These developments can be grouped into five areas, namely:

(i) Modification and selection of parameters,

(ii) Mutations of particles' positions,

(iii) Swarm initialization,

(iv) Hybridization with other techniques, and

(v) Topological structure.

Brief overviews of these are provided below.

2.4.3.1 PSO Parameters

Optimization techniques often have parameters that guide its behaviour as is the case with PSO. These parameters have to be set by the user to achieve good performance. The implication is that different choices of these parameters can cause the technique to perform badly or very well in solving particular problems. Thus, the PSO is a parameter-sensitive technique and selecting good parameters is significant and very challenging [21, 84, 103, 132].

The researches that fall into this category relate to the inertia weight parameter, maximum velocity, constriction factor, acceleration coefficients, random factors and swarm size. These are briefly reviewed below:

(a) Inertia weight parameter and its variants

To further enhance the performance of the PSO, the Inertia Weight Strategy (IWS) was introduced with the aim of enhancing its exploitation and exploration characteristics. This parameter, commonly represented as ω, was originally introduced into the PSO by [104] as a static (constant) factor with a fixed value throughout the execution of the algorithm. It was introduced to balance the scope of local and global searches of PSO and reduce the importance of (or eliminate) velocity clamping during the optimization process [30, 38, 108]. This parameter was added to the velocity updating formula to modify equation (2.1) resulting in equation (2.4).

$$\vec{v}_i^{t+1} = \omega\vec{v}_i^t + c_1\vec{r}_1(\vec{p}_i - \vec{x}_i^t) + c_2\vec{r}_2(\vec{p}_g - \vec{x}_i^t) \quad (2.4)$$

The inertia weight parameter added to the first term at the right hand side of equation (2.4) determines the proportion of the previous velocity that is contributed to the current particles' velocity. This implies that, if the value is high, the velocity is increased and if the value is low the velocity decreases thus a determinant of the speed of the particles.

Over the years several inertia weight strategies have been proposed to dynamically adjust its value at each iteration (see for example [4, 41, 55, 64, 69, 72, 102, 118]). These strategies include random, time varying, chaotic and adaptive inertia weight strategies. These inertia weight strategies have been experimentally proven to enhance the performance of the PSO with varying degrees of success. These variants are briefly discussed below.

i. Random inertia weight strategies

Diversification (exploration) is vital in locating the area of global solution to an optimization problem. This activity can be facilitated mostly by means of randomization. As a result, randomness has been introduced to the IWS by different researchers [36, 47]. This strategy does not have any feedback parameter. The inertia weight thus takes different values randomly assigned at each iteration from a specified interval. In [37] it was empirically discovered that random inertia weight strategy increases convergence in the PSO and could find good solutions to most functions.

ii. Time varying inertia weight strategies

In this category, the value of the inertia weight is computed based on the iteration number. Variants in this category can be broadly divided into two classes namely, linear and nonlinear. The linear time-varying strategies can be further categorized into *linear time decreasing* and *linear time increasing strategies*. The linear time decreasing strategy uses an initially large inertia weight (usually 0.9) which is linearly decreased to a small value (usually 0.4) [29, 41, 71, 72, 113, 129]. There are cases where values other than 0.9 or 0.4 are used [39, 64, 68, 101]. The linear time increasing strategy increases the inertia weight linearly from a specified small value to a final large value [128, 129].

Similarly, the nonlinear time-varying strategies can be categorized into *nonlinear time decreasing* and *nonlinear time increasing strategies*. The nonlinear time decreasing strategy decreases an initially large value nonlinearly to a small value [6, 56, 67, 116]. This allows shorter time for exploration than the linear decreasing methods which spend more time on refining solutions (i.e. exploitation) [29, 67]. These methods seem more appropriate for smoother search spaces [29]. Conversely, the nonlinear time increasing strategy works in the reverse order of the nonlinear time decreasing strategy [88].

A common characteristic of these inertia weight strategies is that the inertia weight computed is bounded by two values which are always pre-defined by the user. These values are the initial value ($\omega_{\min}$) and final value ($\omega_{\max}$) for the increasing strategies. However, for the decreasing strategies, the initial value is $\omega_{\max}$ while the final value is $\omega_{\min}$. With these static values, no room for flexibility is created for the inertia weight computed values.

iii. Chaotic inertia weight strategies

Chaos optimizations have been applied to different aspects of PSO by various researchers over time [31, 27, 46, 45, 70]. The important role of randomization can be understood using the chaos theory. Chaos is mathematically defined as randomness generated by a simple deterministic system [110]. It is generally characterised by three dynamic properties namely, ergodicity, stochastic and sensitivity to its initial conditions [27, 110] which is believe to enhance the search ability of PSO. This seems to be the motivation behind the introduction of chaos feature into IWS by [41] which led to improved optimizing capabilities of the Chaotic Descending Inertia Weight PSO (CDIW-PSO) and Chaotic Random Inertia Weight PSO (CRIW-PSO) due to better global search mobility, convergence speed and convergence precision compared to the Linear Decreasing Inertia Weight PSO (LDIW-PSO) and Random Inertia Weight PSO (RIW-PSO) respectively. There are several other chaotic maps such as the logistic chaotic map which can be used in conjunction with the IWSs to improve the performance of PSO.

iv. Adaptive inertia weight strategies

The adaptive IWSs were also introduced to improve the performance of PSO. These can be grouped into two namely, the fuzzy adaptive inertia weight, which is dynamically adjusted based on fuzzy sets and rules in each iteration [77, 105] and non-fuzzy adaptive inertia weights, which are dynamically adjusted based on some feedback parameters like swarm particle fitness, particle rank, distance to particle, global best positions, and particle success rate [77].

(b) Introduction of maximum velocity

The velocity of a particle as given in equation (2.1), without restriction, can grow unbounded while the particle oscillates around an optimum, increasing its distance to the optimum in each iteration. This initiated the introduction of the velocity clamping effect (or maximum velocity, V_{max}) to avoid velocity divergence. This idea was

introduced by Eberhart and Kennedy in 1995 [36, 103]. It improves the performance of the PSO as it helps particles take reasonably sized steps raking through the search space rather than bouncing and continuously searching outside the solution space. Velocity limits has been widely used in experimental studies [102]. However, efforts have been made to eliminate the use of V_{max} although, researches have shown that velocity clamping has become a standard feature of the PSO [40].

The maximum velocity bounds for particles could negatively affect the performance of the PSO algorithm if it is not properly set. As a result, various works have sought to determine the velocity limits of particles that help to improve the performance of PSO [102, 132]. The three major methods for computing the velocity clamping (V_{min} and V_{max}) in literature are: *(i)* multiplying the search space range with a certain percentage (δ), i.e. $V_{max} = \delta(X_{max} - X_{min})$ and $V_{min} = -V_{max}$ [40], *(ii)* multiplying both the minimum and maximum limits of the search space separately with a certain percentage (δ), i.e. $V_{max} = \delta(X_{max})$ and $V_{min} = \delta(X_{min})$ [132], and *(iii)* assigning the search space upper limit to V_{max}, (δ), i.e. $V_{max} = X_{max}$ [38, 122]. It can be seen from (i) and (ii) that the parameter δ is very important. As a result, different values have been used by different authors [40, 68, 101] for $\delta \in (0,1]$ to determine velocity clamping for particles. In literature, irrespective any of the three methods used, the velocity limits remain constant throughout the lifetime of the algorithm.

From equation (2.2), it is obvious that the velocity of a particle dictates the particle's trajectory and is the direct determinant of its step sizes. Thus, the velocity limit plays important roles in the exploration and exploitation ability of the PSO algorithm, though its selections may be problem-dependent [103]. There exists the possibility of encountering certain practical problems as a result of lack of knowledge regarding the selection of V_{max} leading to the use of a trial-and-error approach in order to make a selection which could be very arduous and time consuming. Allowing the velocity threshold to remain static, either by assigning to it a predefined constant value or a search space threshold, throughout the lifetime of the algorithms, can make the particles have some step size causing them to do more than enough exploration or insufficient exploitation.

(c) Introduction of the constriction factor

To ensure convergence, a PSO with constriction coefficient was proposed by [30] which help transform the velocity update formula in equation (2.1) to that of equation (2.5) below. The introduction of this parameter was to eliminate the need for velocity limit as it is believed to limit the exploration of PSO [38]. However, empirical studies in [38] shows that the constriction factor performed better when used with velocity limit parameter. A mathematical argument presented in [15] revealed that the inertia weight model is equivalent to the constriction factor. Earlier on in [38], the two parameters were observed to be the same and the PSO with the constriction factor was considered to be a special case of an algorithm with inertia weight. Another study reported in [132] shows that the constriction factor PSO has varied efficiencies relative to unimodal and multimodal problems being solved. However, based on the findings of [15], it is not necessary to compute the constriction factor using equation (2.5) because the sum of the learning coefficients which is required to be greater than 4 produces an unnecessary oscillation of the particles.

$$v_i^{t+1} = \chi\left(v_i^t + c_1 r_1(\vec{p}_i - x_i^t) + c_2 r_2(\vec{p}_g - x_i^t)\right) \tag{2.5}$$

where,

$$\chi = \frac{2\kappa}{\left|2 - \quad - \sqrt{\left(\quad - 4\right)}\right|}$$

with

$$\phi = c_1 r_1 + c_2 r_2$$

and

$$\phi > 4$$

The parameters $v_i, x_i, c_1, c_2, r_1, r_2, \vec{p}_i$ and $\vec{p}_g$ are as defined in Section 2.4.1. Parameter χ is known as the constriction factor which is a function of $c_1$and c_2; $\kappa \in [0,1]$ is an arbitrary constant that is used to adjust the value of χ.

(d) Acceleration coefficients

These parameters are positive values that are commonly represented as c_1 (cognitive scaling parameter) and c_2 (social scaling parameter). They regulate the relative velocity of each particle towards the local and global best respectively. The values of 2.0 as originally assigned to these parameters in the OPSO [63] have been adopted by many researchers over the years [4, 29, 39, 41]. As a result of the sensitive roles of

these parameters in the performance of PSO, other researchers have attempted to adjust them through empirical studies [50, 95-107]. In [50], the role of the acceleration coefficients on the performance of PSO was investigated by using unsymmetrical transfer range of acceleration coefficients. Simulation studies exhibited an improved optimum solution for most of the benchmarks used especially when changing c_1 from 2.75 to 1.25 and c_2 from 0.5 to 2.25, over the full range of the search [50].

Furthermore, in [95], the New PSO (NPSO) was proposed. Here, the cognitive acceleration coefficient c_1 was split into good experience component c_{1g} and bad experience component c_{1b} to help the particles move towards their previous best positions and away from their previous worst positions in order to facilitate exploration capability. Similarly, the Anti-Predatory PSO (APSO) was proposed by [96], where the cognitive acceleration coefficient c_1 was split into good experience component c_{1g} and bad experience component $c_{1b.}$ c_2 was also split into the good experience component c_{2g} and the bad experience component c_{2b}. The bad experiences help particles to by-pass their previous worst positions while good experiences help particles move towards their previous best positions. Likewise in [107], the Time-Varying Acceleration Coefficients PSO (PSO-TVAC) was introduced to enhance the global search in the early part of the optimization and to encourage particles' convergence towards the global optimum at the end of the search. This was achieved by linearly decreasing the cognitive parameter c_1 from a high value c_{1max} to a low value c_{1min} while the social parameter, $c_{2,}$ was linearly increased from a low value c_{2min} to a high value of c_{2max}. Discussions on other strategies for determining these acceleration coefficients can be found in [57].

From the preceding information regarding acceleration coefficients, it is clear that in a bid to make the PSO technique perform better, some complexities and extra computational efforts have been added to or introduced to the technique. If the activities of these parameters could be compensated for, with less effort and complexities, in other parts of the PSO technique, then the parameters could be removed from the velocity updating formula to avert the extra computational complications.

(e) Random factor

A closer look at the PSO algorithm reveals that randomness plays a very useful role in making the algorithm while seeking effective solution to optimization problems. Randomness comes into play at the point of initializing the particles in the solution space and in updating the velocities of particles at each iteration of the algorithm. The presence of random factors in the velocity formula enhances stochastic tendency and slows down convergence in order to promote the state space exploration and prevent premature convergence to non-optimal points [116]. This random feature has contributed immensely to the performance of PSO [41, 46].

(f) Swarm size

This is often set relative to the dimensionality and perceived complexity of a problem. Values in the range 20-50 are common [20, 63, 86, 117, 122], depending on the problem being solved. The convergence of PSO can also be influenced by the swarm size. Small size of swarm results in fewer numbers of function evaluations and consequently faster clock time, but in most cases, a large number of algorithm iterations is needed while large swarm size requires more function evaluations and fewer numbers of iterations [113, 132]. Tuning this parameter is seen to be of minor importance [98], thus, there appears to be no generally defined swarm size in the literature.

2.4.3.2 Mutation operators

In order to increase the diversity of the swarm and to prevent premature convergence to local optimal, various mutation operators have been introduced to the PSO [5, 31, 46, 68, 69]. Chaos mutation operator based on logistic map was used by [27, 46] and another based on Zaslavskii was used by [31]. In [45], twelve different chaos maps were implemented to tune the attraction parameter of an accelerated PSO algorithm.

2.4.3.3 Swarm initialization

Swarm initialization involves the way particles are randomly distributed in the search space at the initial stage relative to their positions and velocities, before the algorithm starts execution. There are two sides to this, i.e. the random number generator used and the way particles are distributed, both of which could enhance the computational behaviour of PSO technique during the search process.

Random number generators: These are systems with the ability to generate sequences of random numbers according to a probability function [16]. Different types of random number generators have been experimentally implemented to initialize particles in the search space [16, 57, 81] but their efficiency seems to be problem specific, as a certain initialization technique may lead to desirable behaviour in one problem and undesirable in another [57].

Distribution of particles: During initialization, particles could be distributed symmetrically or asymmetrically. When particles are distributed within the entire feasible search space, with the global optimum lying within the space, most especially when it is at the centre of distribution, it is said to be symmetrical; this is common among the PSO variants relative to the benchmarking problems [81]. The initialization is asymmetric when the particles are distributed within a subspace of the entire feasible search space that does not contain the global optimum. The latter method is referred to as region scaling and is most applicable as a research standard for performance testing and comparison of algorithms when both the problem and its optimum are known [20].

Most PSO variants use uniformly distributed random numbers for the initialization of particles [81]. However, the random number generator used to initialize swarm in PSO is not commonly specified by researchers in literature. This is not an encouraging practice because it makes performance comparisons of PSO variants difficult.

2.4.3.4 Swarm topological structure

This is the communication structure by which all the particles in the swarm are organized to share information with each other when they are searching for solution in the search space. It typically consists of bidirectional links connecting pairs of particles and the best point found by any particle affects its neighbourhood. For further information on swarm topological structure and the various ways they are categorized can be found in [20, 57, 86, 94]. The type of topology used to implement PSO can affect its efficiency and could be problem dependent.

2.4.3.5 Hybridization with other techniques

Hybridization is the combination of two or more techniques, taking advantage of their strengths, to build up a better technique that will be of more benefit compared with the original individual techniques. Two popular ways of hybridization are sequential and parallel hybridizations [44, 111]. Using these methods, the PSO has been hybridized with different population-based techniques over the years. In the research carried out by [111], hybridization with the Genetic Algorithm is the most popular choice among researchers followed by Differential Evolution and Ant Colony Optimization algorithms. Other techniques that have been combined with the PSO are bacterial foraging optimization [14, 71] and simulated annealing [6, 51, 58, 106]. In [70], PSO was hybridized with a chaotic local search procedure based on logistic map. The logistic and tent chaotic maps were respectively used as inertia weights by [27] in binary PSO to handle the feature selection problem.

PSO has also been hybridized with other local search techniques to help strengthen its local search ability. Among these are Hill Climbing [59], Golden ratio method [127], Local interpolation search [114], Adaptive local search [58] and the Quasi-Newton method [130]. Some other local search techniques hybridized with the PSO are reviewed in [94, 111].

Chapter 3

Studies based on Dynamic Velocity and Search Space Limits

The velocity limits (threshold) as well as the search space limits play important roles in the efficiency of the PSO technique. They are used to control the extent of the movements of particles when searching for (near) optimal solutions to optimization problems within the search space. This control helps particles not to move out of the search space of the problem, thereby forcing them to concentrate on the environment where solutions to the problems can be found. The velocity limits are generally represented by V_{min} and V_{max} to form an interval $[V_{min}, V_{max}]$, where V_{min} is the minimum velocity and V_{max} is the maximum velocity of particles in the search space.

These parameters are often pre-set by users when implementing PSO. The search space limits define the boundaries for the decision variables of the problems being optimized and the dimensions of the variables are expected to take values from within this space defined by the boundaries. The search space limits are generally represented by X_{min} and X_{max} to form an interval $[X_{min}, X_{max}]$, where X_{min} is the minimum value and X_{max} is the maximum value the decision variables can obtain relative to the search space. These parameters are also often pre-set by users when implementing the PSO and they vary with the type of optimization problems. This chapter presents two research articles (Paper 1 and Paper 2) based on studies on the velocity and search space limits of PSO. Furthermore, the chapter reports results another PSO variant based on greedy and adaptive methods of obtaining the global swarm best.

3.1 Paper 1: On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global optimization

In Paper 1, the effects of different velocity limits on the performance of PSO were studied and some of the values obtained for the limits were used to improve the

performance of one of the PSO variants in literature, that is, the LDIW-PSO. This variant has been considered by some researchers to be less effective compared to their respective proposed PSO variants with numerical evidences recorded in literature. In trying to validate these claims, several numerical simulations were performed using the improved LDIW-PSO. Empirical results obtained showed that LDIW-PSO performed better than these variants. Compared to other recent PSO variants with different inertia weight strategies on the same test problems, it was also discovered that LDIW-PSO had a competitive advantage. The findings in Paper 1 revealed that previous claims of its inferior performance might have been due to some unfavourable experimental settings. With good experimental settings, LDIW-PSO will perform competitively well compared to many PSO variants. Further simulation results that can provide useful hints for deciding the setting velocity limits for particles for LDIW-PSO were provided in the paper.

3.2 Paper 2: Improved Particle Swarm Optimizer with Dynamically Adjusted Search Space and Velocity Limits for Global Optimization

Based on the positive effects of velocity limits on the performance of the PSO technique in Paper 1, it became necessary for further studies on velocity and search space limits. This is the focus of Paper 2. This further study was motivated by the original goal of PSO of finding solutions to optimization problems much faster than traditional methods. Also, spending time to find optimal settings for the velocity limits parameters could count against any superiority claim over competing methods. Another motivation for further studies hinges on two major features that characterize many of the PSO variants in literature namely, the static particle search space and velocity limits. That is, once values for these parameters are set, they remain the same throughout the lifetime of the algorithm. This has limited the flexibilities of these variants in obtaining optimal solutions for many of optimization problems. Paper 2 therefore studied the OPSO with the aims of improving its performance and compare results thereof with some efficient PSO variants recorded in literature.

Instead of using the inertia weight parameter, which is the common tool being used to address the problem of premature convergence associated with PSO, Paper 2 worked directly with the velocities of the particles. This is because the velocities of particles

are the direct determinants of the particles' step sizes. The velocity limits were made to vary throughout the lifetime of the algorithm to create opportunities for the algorithm to obtain better quality solutions to optimization problems. Also, the solution search spaces were made to vary to prevent particles from spending unnecessary time searching areas that may not be necessary in finding good solutions.

Numerical simulation results show that the improved OPSO is very consistent in convergence velocity, convergence accuracy, global search ability and robustness than all the PSO variants adopted for comparisons. The findings in Paper 2 further revealed that if the velocity limits and solution search space of particles are allowed to vary dynamically relative to the values of particles' dimension, there is likelihood of great improvement in the performance of the algorithm. This results from the better exploration and exploitation activities of the algorithm with added flexibility in concentrating on the promising areas in the solution search space for further search by the particles instead of the entire space all the time.

3.3 Particle Swarm Optimizer based on greedy and adaptive methods

This is an additional work, not yet reported or submitted as an article to any journal or conference as at the time of this thesis. The study attempt to further improve the way global best is obtained from the personal bests of all the particles in the PSO technique. In this variant, adaptive feature was introduced in the process of obtaining the global best. Also, the way velocity limits and search space limits were obtained is different from the methods mentioned in section 3.2 and reported in Paper 2. This variant is named PSO based on greedy and adaptive methods (GAPSO).

3.3.1 Motivation

This study was motivated by the fact that:

i. PSO uses pure greedy method in searching for (near) optimal solution. We seek ways to complement this with an adaptive strategy for better efficiency.
ii. We also observed that premature convergence to a non-global local minimum is more likely to occur with a greedy strategy in handling multimodal problems which might not be of much problem if an adaptive approach is adopted.

In GAPSO, four phases are involved in the determination of the global best position from the personal bests of particles as well as in the calculation of the search space and the velocity limits.

Phase 1: *Personal best construction*

In this phase, the personal best of each particle is obtained in the normal way as in the OPSO by using greedy method. That is, if the current position of a particle is better than the best it last visited, it is retained otherwise it is replaced with the best last visited position. With this, a vector of personal bests $\vec{P} = (p_1, p_2, \cdots, p_n)$ is created for all the particles, where n is the swarm size.

Phase 2: *Swarm splitting phase*

After obtaining personal bests ($\vec{P}$) for all the particles, a threshold is defined using a value-based method. In the value-based method, the parameter $\alpha \in [0,1]$ is used in defining the threshold. Assume g is the candidate evaluation function which maps every elements c_i of the set of yet to be added particles C to a real value, $g_{min} = \min\{g(c_i)\}, \forall c_i \in C$ and $g_{max} = \max\{g(c_i)\}, \forall c_i \in C$. Since minimization problems are considered, all particles which have objective function value smaller than the threshold $\mu = g_{min} + \alpha(g_{max} - g_{min})$ are included in *group 1* while the other particles are included in *group 2*. Thus, the objective function value of each particle must be in the interval $g(c_i) \in [g_{min}, \mu]$ to be included in *group 1*. If $\alpha = 0$ the selection is greedy, but purely random if $\alpha = 1$.

Listed below are three different approaches of choosing α,

i. Choosing α randomly from a uniform discrete probability,
ii. Choosing α from a non-uniform decreasing discrete probability, and
iii. Fixing α to a value close to the purely greedy choice.

Approach (i) is currently used in the implementation of GAPSO.

This process of obtaining the personal bests involves using a *greedy* method whereas an *adaptive* method is used in the algorithm to update the particles in *group1*. As long as this phase continues, the solution found gradually improves.

Phase 3: *Obtaining global best position*

Instead of obtaining the global best by picking the best of all personal bests, it is obtained by collecting the least value in each dimension across all particles in *group 1*. Thus, the global best position ($\overrightarrow{gBest}$) for entire swarm is obtained using equation (3.1).

$$gBest^j = \min_i(x_i^j) \tag{3.1}$$

where, i = 1,2 ,…, n, is particle's index, j = 1,2 ,…, d, is the index of particle's dimension, n is the swarm size and d is the dimension size.

Phase 4: *Obtaining velocity and search space limits for particles*

The velocity and search space limits are obtained using *group 2* of particles as follows:

(i) During each iteration, the largest dimensions value (L_d) and the smallest dimension value (S_d) among the dimensions of all the particles, are obtained according to equations (2) and (3), where, x_i^j is the i^{th} particle with j^{th} dimension.

$$L_d \leftarrow \max_i\left(\max_j(x_i^j)\right) \tag{3.2}$$

$$S_d \leftarrow \min_i\left(\min_j(x_i^j)\right) \tag{3.3}$$

(ii) The upper limit x_{max} and lower limit x_{min} of the solution search space for the particles were obtained according to equations (3.4) and (3.5), where | **.** | means absolute value.

$$x_{max} \leftarrow max(|L_d|, |S_d|) \tag{3.4}$$

$$x_{min} \leftarrow -x_{max} \tag{3.5}$$

(iii)After obtaining x_{max} and x_{min}, they are used to compute the upper (v_{max}) and lower (v_{min}) particle velocity limits as defined in equations (3.6) and (3.7).

$$v_{max} \leftarrow \mu x_{max} \tag{3.6}$$

$$v_{min} \leftarrow \mu x_{min} \tag{3.7}$$

where, μ is a velocity clamping percentage which is used to scale the upper and lower solution space limits to help reduce the velocity range for particles.

Equation (3.8) was used to update the positions of particles at each iteration.

$$x_i = \begin{cases} x_{min} + (x_{min} - x_i) \times random(0,1) & \text{if } x_i < x_{min} \\ x_{max} - (x_i - x_{max}) \times random(0,1) & \text{if } x_i > x_{min} \end{cases} \tag{3.8}$$

Simulation experiments were conducted to implement this new algorithm (GAPSO) in order to determine its weakness and strength, to guide the direction for its further improvement(s).

3.3.2 Experimental settings

Since PSO is a stochastic algorithm, all experiments were repeated 50 times with different random seeds. The performance of each approach takes into account the Mean Best Fitness (MBF), Standard Deviation (SD), Success Rate (SR), number of Function Evaluations (FE) to satisfy the success criteria and Average Computer Time (ACT) in minutes for all the 50 runs. The proposed PSO variant was applied to 31 benchmark test problems in Appendix A as obtained from literature [8, 22, 60, 91, 92].

The number of variables (dimensions) for all functions in the experiments ranges from 2 to 30. Swarm size was set to 30; maximum number of iteration was set to 5,000 for dimensions of 30 and 2,000 for others. The stopping criterion is to allow algorithms run for the maximum number of iterations. For the test problems that have their global minimum as zero (0.0), a run was considered successful if at the end of maximum iteration the algorithm obtains a result less than 10^{-5}. For other test problems, a run was considered successful if at the end of maximum iteration the algorithm obtains a result less than the success criteria stated in Table 3.1. The parameters c_1 and c_2 were set to 2.0. For LDIW-PSO, $V_{\max}$ was set relative to the search space of each problem, using $V_{\max} = 0.05 \times X_{\max}$, where $X_{\max}$ is the maximum value of the domain of x. This $V_{\max}$ setting was used because it greatly increased the efficiency of LDIW-PSO. For GAPSO, at the beginning $V_{\max} = X_{\max}$, but $V_{\max}$ was subsequently adjusted using Equations (3.6) and (3.7).

Table 3.1: Success criteria for some of the test problems

Test Problem	ALFP	CML6	CSM1	EXPN	ESOM	HTMF	MCLZ	ROSB
Success criteria	-0.3520	-1.0315	-2.9999	-0.9999	-0.9999	-3.85	-1.8012	30.0

3.3.3 Experimental results and discussions

Presented in Table 3.2 are the results obtained from the experiments when LDIW-PSO and GAPSO were used on the benchmarked test functions. The bold values show better optimal results. From the results, it is clear that LDIW-PSO generally performed better in low dimension problems while GAPSO generally performed better in high dimension problems. However, the two algorithms performed equally in BKY2 and EXPN problems. In these two problems, GAPSO executed fewer number of FEs with slightly higher ACT. The higher ACT is perhaps as a result of the time used by GAPSO to compute Equations (3.1) – (3.7). A general observation is that, GAPSO has the ability to escape local optima than LDIW-PSO in complex search spaces.

Observations from the performance of the two algorithms might suggest the need to hybridize them with other technique for improved overall performance. GAPSO needs further improvements to make it perform well on low-dimensional problems. These might include:

i. Introduction of randomness into equation (3.1). This will involve a random selection of some personal bests of particles in *group 1* to obtain the global best.
ii. Implementing other approaches of choosing α
iii. Introduction some mutations to the positions of particles.

Table 3.2: Results obtained for the test problems using LDIW-PSO and GAPSO

Test Prob.	LDIW-PSO						GAPSO					
	MBF	SD	midERR	SR	AFE	ACT	MBF	SD	midERR	SR	AFE	ACT
ACKL	1.6289e-14	4.6707e-15	1.4655e-14	100	105595	1.43	**8.7574e-15**	2.6213e-15	7.5495e-15	100	32101	2.04
ALFP	**-3.5239e-01**	3.3307e-16	8.6074e-05	100	603	0.04	-3.5131e-01	1.8381e-03	3.8840e-04	42	87820	0.05
BEAL	**3.0483e-02**	1.4933e-01	0.0000e+00	96	4728	0.05	8.2188e-01	4.2357e-01	9.3484e-01	0	62000	0.05
BELA	**0.0000e+00**	0.0000e+00	0.0000e+00	100	3287	0.03	5.4024e-02	1.5026e-01	1.8151e-02	0	62000	0.04
BKY1	**0.0000e+00**	0.0000e+00	0.0000e+00	100	8909	0.03	1.5543e-16	8.5998e-17	2.2204e-16	100	1882	0.05
BKY2	0.0000e+00	0.0000e+00	0.0000e+00	100	8546	0.04	0.0000e+00	0.0000e+00	0.0000e+00	100	1923	0.03
BOOT	**0.0000e+00**	0.0000e+00	0.0000e+00	100	4063	0.03	7.3119e-01	7.1674e-01	4.9647e-01	0	62000	0.04
BRWN	1.3800e+01	2.0778e+01	1.9478e-34	62	168093	4.56	**6.8330e-48**	3.5987e-47	2.2084e-51	100	20259	4.45
CML3	2.2116e-148	1.3789e-147	2.99172-152	100	1516	0.05	**4.9407e-324**	0.0000e+00	0.0000e+00	100	3354	0.07
CML6	**-1.0316e+00**	2.2204e-16	1.0316e+00	100	58	0.05	-1.0250e+00	8.1582e-03	4.1526e-03	6	948720	0.06
CIGR	4.8951e-29	1.2908e-28	1.7227e-30	100	100744	1.24	**7.6967e-46**	4.0746e-45	7.0415e-49	100	30658	1.39
CSM1	-2.6158e+00	2.2510e-01	4.4335e-01	6	2425340	1.38	**-3.0000e+00**	0.0000e+00	0.0000e+00	100	20957	1.49
CVLE	**2.4795e-01**	1.2575e+00	7.0065e-04	4	1483260	0.08	9.2771e+00	4.9589e+00	9.0742e+00	0	62000	0.10
DEJ4	1.6220e-42	5.5364e-42	2.5449e-44	100	69724	1.21	**9.8669-62**	5.2637e-61	2.6681e-66	100	20242	1.40
DIXP	**6.6667e-01**	3.0927e-16	6.6667e-01	0	150000	2.03	9.5727e-01	1.2447e-02	9.5819e-01	0	155000	2.25
EXPN	-1.0000e+00	4.7103e-17	1.1102e-16	100	58001	1.27	-1.0000e+00	8.7419e-17	1.1102e-16	100	10670	1.42
ESOM	**-1.0000e+00**	0.0000e+00	0.0000e+00	100	1543	0.03	-9.7990e-01	6.6969e-02	3.9320e-03	4	1457015	0.05
GWNK	1.5137e-02	1.8336e-02	9.8610e-03	32	415785	1.52	**0.0000e+00**	0.0000e+00	0.0000e+00	100	30870	1.54
LVM1	**1.4514e-02**	3.5972e-02	1.6995e-32	86	95692	2.23	1.3511e-02	6.5184e-02	2.2328e-04	0	155000	2.32
LVM2	**6.5924e-04**	2.6094e-03	3.0753e-32	94	85950	2.16	9.8279e-03	1.7673e-02	6.7619e-03	0	155000	2.36
HTMF	-3.7746e+00	6.6502e-02	7.4606e-02	4	1440360	0.23	**-3.7898e+00**	1.1711e-01	3.1946e-02	30	144474	0.21
MTYS	6.1019e-117	3.8018e-116	3.2015e-120	100	1166	0.03	**1.4880e-247**	0.0000e+00	2.6758e-272	100	3262	0.03
MCLZ	**-1.7415e+00**	3.8134e-02	5.2467e-02	0	60000	0.05	-1.6893e+00	1.6901e-01	3.8400e-02	0	62000	0.07
NQTC	1.4648e-03	5.2335e-04	1.4087e-03	0	150000	1.28	**4.4310e-04**	3.4823e-04	3.8526e-04	2	7423760	1.41
NCRA	3.6641e+01	1.1758e+01	3.6501e+01	0	150000	1.49	**6.0002e-01**	4.2001e+00	0.0000e+00	98	66826	2.14
PLZ1	**4.8012e-32**	2.1277e-31	6.7335e-33	100	80986	2.32	8.9313e-05	4.3428e-05	8.8466e-05	0	155000	2.30
PLZ2	**8.7899e-04**	2.9808e-03	4.8010e-32	92	101760	2.39	4.7446e-03	2.2900e-03	4.3059e-03	0	155000	2.49
PRDC	9.1600e-01	3.6661e-02	9.0000e-01	0	60000	0.05	**9.0000e-01**	8.8818e-16	9.0000e-01	0	60175	0.07
RAS1	3.1873e+01	1.1360e+01	2.8831e+01	0	150000	1.47	**8.9601e-07**	6.2721e-06	0.0000e+00	98	83736	1.55
RAS2	3.0505e+01	1.0002e+01	3.0844e+01	0	150000	1.37	**1.2574e-03**	8.8017e-03	0.0000e+00	98	88057	1.59
ROSB	3.1898e+01	2.1961e+01	2.3236e+01	86	102257	2.43	**2.8696e+01**	9.2685e-04	2.8696e+01	100	19495	2.56

INCLUDED ARTICLES

PAPER 1

ON THE PERFORMANCE OF LINEAR DECREASING INERTIA WEIGHT PARTICLE SWARM OPTIMIZATION FOR GLOBAL OPTIMIZATION [11]

M. A. ARASOMWAN AND A. O. ADEWUMI

The Scientific World Journal, Volume 2013, Article ID 860289, 12 pages

http://dx.doi.org/10.1155/2013/860289

PAPER 2

IMPROVED PARTICLE SWARM OPTIMIZER WITH DYNAMICALLY ADJUSTED SEARCH SPACE AND VELOCITY LIMIT FOR GLOBAL OPTIMIZATION

M. A. ARASOMWAN AND A.O. ADEWUMI

Submitted to International Journal of Artificial Intelligence Tool

Chapter 4

Studies based on Swarm Success Rate and Chaotic Maps

The drive to further enhance the performance of the PSO technique led to the introduction of the inertia weight parameter into the PSO in 1998 [104] to balance its intensification and diversification activities. Intensification (exploitation) searches for the current best solutions and selects the best candidate; while diversification (exploration) allows the algorithm explore the search space more efficiently mostly by means of randomization to locate promising regions that would proffer better solutions. Motivated by the possibility of increasing the search ability of PSO with chaotic optimization, the Chaotic Descending Inertia Weight PSO (CDIW-PSO) and Chaotic Random Inertia Weight PSO (CRIW-PSO) were introduced in [41]. These variants used logistic chaotic map, to improve the performances of the two PSO variants that implemented two pioneering inertia weight strategies: linear decreasing and random inertia weight strategies.

Chaos is mathematically defined as randomness generated by a simple deterministic system. Also, the swarm success rate was embedded in the inertia weight strategy as a feedback parameter in [77] to enhance the performance of the PSO technique. Other chaotic maps different from the existing ones but relative to the inertia weight strategies, were utilised in this chapter. Moreover, the swarm success rate was applied in a different way to that employed in [77].

In the three papers (Paper 3, Paper 4 and Paper 5) included in this chapter, three different types of studies were carried out with three major goals as follows:

i. To investigate the effects of various chaotic maps in comparison with the logistic map when used in the inertia weight strategy,
ii. To propose new inertia weight strategies based on swarm success rate combined with the logistic map on one hand and based on only swarm success rate on the other hand.

iii. To use the proposed variants to further improve the effectiveness of the PSO algorithms in terms of convergence speed, global search ability, robustness and increased solution accuracy.

4.1 Paper 3: On Adaptive Chaotic Inertia Weight in Particle Swarm Optimization

In Paper 3, two adaptive chaotic inertia weights which combine the swarm success rate feedback parameter with the logistic chaotic mapping to harness the adaptive and chaotic characteristics of the individual techniques are proposed.

4.2 Paper 4: An Improved Particle Swarm Optimizer based on Swarm Success Rate for Global optimization Problems

Based on the findings in Paper 3, the swarm success rate was found to be a very useful tool for enhancing the performance of PSO. Paper 4 further explored the potentiality of inertia weight combined with the swarm success rate as the latter provide useful information about the particles in the search space. It was ascertained from literature that many of the inertia weight strategies which originated from the LDIW strategies always have fixed initial and final values of inertia weight with the exception of CDIW-PSO and CRIW-PSO which utilize chaotic values to adjust part of the boundaries.

Paper 4 proposes two new PSO variants namely, the Swarm Success Rate Decreasing Inertia Weight PSO ($SSRDIW_{PSO}$) and Swarm Success Rate Random Inertia Weight PSO ($SSRRIW_{PSO}$). It is believed that these variants performed better because the swarm success rate which served as a feedback parameter helped in realizing the state of the swarm in the search space and hence, adjusted the value of the inertia weight in each iteration appropriately for better results compared with when chaotic value which has no information about the state of the swarm is used.

More experiments were conducted using 31 test problems (see Appendix A) to further test the performance of $SSDIW_{PSO}$ compared to LDIW-PSO and CDIW-PSO. The dimension of the test problems ranges from 2 to 30. Other experimental settings used

in Paper 4 were used. Presented in Tables 4.1 – 4.3 are the results obtained implementing the three variants. Best results obtained among the three variants are indicated in bold. In all the results, the average performance of $SSRDIW_{PSO}$ in all the performance measurements is better than the competing variants.

Table 4.1: Mean Best Fitness (MBF) and Standard Deviation (SD) for the three PSO variants

Test Problems	MBF			SD		
	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$
ACKL	1.5323e-10	1.6573e-14	**1.3944E-14**	1.6311e-10	1.2494e-14	**3.0972E-15**
ALFP	-3.5239e-01	-3.5239e-01	-3.5239e-01	3.3307e-16	3.3307e-16	3.3307e-16
BEAL	**4.5724e-02**	6.0966e-02	7.6207e-02	**1.8098e-01**	2.0674e-01	2.2862e-01
BELA	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BKY1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BKY2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BOOT	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BRWN	4.0640e+01	**1.4720e+01**	3.8080e+01	3.4792e+01	**2.1188e+01**	3.4597e+01
CML3	3.6188e-148	1.6838e-303	**0.0000e+00**	1.4715e-147	**0.0000e+00**	**0.0000e+00**
CML6	-1.0316e+00	-1.0316e+00	-1.0316e+00	2.2204e-16	2.2204e-16	2.2204e-16
CIGR	8.2652e-29	3.3044e-38	**1.8190e-68**	3.7258e-28	2.3131e-37	**1.1950e-67**
CSM1	**-2.5862e+00**	-2.6305e+00	-2.5951e+00	2.6928e-01	2.1568e-01	**2.4518e-01**
CVLE	8.7573e-04	1.5782e-01	**6.2974e-04**	1.0333e-03	1.1006e+00	**9.6451e-04**
DEJ4	1.3210e-40	8.2972e-83	**2.0480e-120**	9.1181e-40	3.0762e-82	**1.2303e-119**
DIXP	6.6667e-01	6.6667e-01	6.6667e-01	**3.2101e-16**	3.5388e-16	3.3233e-16
EXPN	-1.0000e+00	-1.0000e+00	-1.0000e+00	2.7195e-17	**1.5701e-17**	3.5108e-17
ESOM	-1.0000e+00	-1.0000e+00	-1.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
GWNK	1.6122e-02	1.3480e-02	**9.9941e-03**	1.6529e-02	1.4403e-02	**9.3724e-03**
LVM1	1.2440e-02	1.0367e-02	**8.2935e-03**	3.3688e-02	3.1101e-02	**2.8125e-02**
LVM2	**2.6235e-31**	8.7899e-04	4.1449e-03	**1.0428e-30**	2.9808e-03	1.4021e-02
HTMF	-3.7768e+00	-3.7753e+00	**-3.7500e+00**	**5.5431e-02**	5.9576e-02	1.5927e-01
MTYS	8.2243e-116	1.9156e-213	**8.0597e-274**	5.3343e-115	**0.0000e+00**	**0.0000e+00**
MCLZ	**-1.7576e+00**	-1.7654e+00	-1.7675e+00	3.6620e-02	2.3939e-02	**1.5048e-02**
NQTC	**1.4412e-03**	2.3647e-03	3.6301e-03	**6.0783e-04**	7.8462e-04	1.3296e-03
NCRA	**3.9561e-01**	3.9921e+01	4.0406e+01	**1.0911e+01**	1.1701e+01	1.1733e+01
PLZ1	1.8548e-32	**9.4183e-33**	1.4329e-02	**4.9741e-32**	1.5041e-31	1.0031e-01
PLZ2	1.0987e-03	**8.7899e-04**	3.0765e-03	3.2962e-03	**2.9808e-03**	4.9333e-03
PRDC	9.1400e-01	9.2000e-01	**9.0800e-01**	3.4699e-02	4.0000e-02	**2.7129e-02**
RAS1	3.1495e+01	**3.1396e+01**	3.2390e+01	9.1559e+00	1.0665e+01	**9.0203e+00**
RAS2	3.3948e+01	**3.2814e+01**	3.4306e+01	1.0491e+01	**9.2322e+00**	1.0594e+01
ROSB	2.9097e+01	**2.5845e+01**	2.9072e+01	1.6590e+01	**1.4162e+01**	2.0483e+01

Table 4.2: Success Rate (SR), Average Function Evaluation (AFE) and Average Computer Time (ACT in minutes for all the runs) for the three PSO variants

Test Prob.	SR (%)			AFE			ACT (min)		
	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$
ACKL	100	100	100	71937	41092	**24071**	**1.06**	1.07	1.09
ALFP	100	100	100	391	**345**	389	**0.04**	0.03	**0.04**
BEAL	**94**	92	90	**5929**	6281	7664	0.05	0.05	**0.04**
BELA	100	100	100	3220	1108	**852**	0.04	**0.03**	**0.03**
BKY1	100	100	100	8868	1934	**1558**	**0.03**	0.04	**0.03**
BKY2	100	100	100	8915	1979	**1550**	0.04	**0.03**	**0.03**
BOOT	100	100	100	3458	1206	**1084**	0.03	**0.02**	0.03
BRWN	28	**64**	28	461863	**112559**	400440	**5.01**	5.07	5.02
CML3	100	100	100	1705	807	**803**	**0.05**	**0.05**	0.06
CML6	100	100	100	1385	745	**695**	0.05	0.05	**0.04**
CIGR	100	100	100	100362	46835	**22705**	**1.3**	1.29	1.32
CSM1	4	**10**	6	3674550	**1369698**	2358920	**1.48**	1.49	1.53
CVLE	6	2	**14**	997040	2970360	**403037**	0.07	0.08	**0.06**
DEJ4	100	100	100	69273	18272	**7420**	1.34	1.37	**1.30**
DIXP	0	0	0	150000	150000	150000	**2.17**	2.24	**2.17**
EXPN	100	100	100	56761	11175	**5107**	1.33	**1.27**	1.31
ESOM	100	100	100	1407	819	**720**	**0.03**	0.04	0.04
GWNK	**36**	30	32	362598	395244	**337328**	**1.53**	2.01	2.01
LVM1	88	90	**92**	92961	37551	**22215**	2.32	**2.25**	2.3
LVM2	**100**	92	78	79616	**42177**	56291	2.4	2.34	**2.25**
HTMF	4	**6**	**6**	1440195	**940110**	940170	**0.19**	0.21	0.20
MTYS	100	100	100	1117	**704**	740	**0.03**	**0.03**	0.03
MCLZ	0	0	0	60000	60000	60000	0.06	**0.05**	**0.05**
NQTC	0	0	0	150000	150000	150000	**1.31**	1.36	1.37
NCRA	78	80	**84**	46263	39569	**30292**	2.03	**1.50**	1.58
PLZ1	**100**	**100**	98	81662	31735	**20038**	2.38	2.37	**2.21**
PLZ2	**90**	**92**	72	105741	**49135**	73847	5.30	2.40	**2.31**
PRDC	0	0	0	60000	60000	60000	0.05	0.05	**0.04**
RAS1	**96**	94	**96**	24691	12599	**8379**	1.54	**1.5**	1.53
RAS2	90	92	**94**	37342	15856	**11955**	2.10	2.09	**2.05**
ROSB	90	**94**	86	90851	**30902**	34723	3.30	3.30	**3.28**

Table 4.3: Mean error (MEANERR), least error (LEASTERR) and median error (MEDIANERR) for the three PSO variants

Test Prob.	MEANERR			LEASTERR			MEDIANERR		
	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$	LDIW-PSO	CDIW-PSO	$SSRDIW_{PSO}$
ACKL	1.5323e-10	1.6573e-14	1.3944E-14	8.1681e-12	7.5495e-15	7.5495E-15	1.1020e-10	**1.4655e-14**	**1.4655E-14**
ALFP	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05	8.6074e-05
BEAL	4.5724e-02	6.0966e-02	7.6207e-02	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BELA	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BKY1	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BKY2	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BOOT	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
BRWN	4.0640e+01	1.4720e+01	3.8080e+01	2.0057e-37	1.8120e-67	1.9220e-91	3.2000e+01	**1.5906e-54**	3.2000e+01
CML3	3.6188e-148	1.6838e-303	0.0000e+00	2.8046e-157	0.0000e+00	0.0000e+00	1.3218e-150	6.9811e-313	**0.0000e+00**
CML6	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05	2.8453e-05
CIGR	8.2652e-29	3.3044e-38	1.8190e-68	5.4450e-33	1.0222e-65	2.9552e-89	4.7527e-30	1.0087e-52	**1.6144e-78**
CSM1	4.1380e-01	3.6946e-01	4.0493e-01	0.0000e+00	0.0000e+00	0.0000e+00	4.4335e-01	4.4335e-01	4.4335e-01
CVLE	8.7573e-04	1.5782e-01	6.2974e-04	3.4675e-07	1.7165e-06	1.1562e-08	6.1549e-04	5.4307e-04	**3.4693e-04**
DEJ4	1.3210e-40	8.2972e-83	2.0480e-120	1.5984e-48	2.7001e-92	3.5283e-131	2.4644e-44	1.4147e-86	**4.4213e-125**
DIXP	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01	6.6667e-01
EXPN	1.1768e-16	1.1324e-16	1.2212e-16	1.1102e-16	1.1102e-16	1.1102e-16	1.1102e-16	1.1102e-16	1.1102e-16
ESOM	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
GWNK	1.6122e-02	1.3480e-02	9.9941e-03	0.0000e+00	0.0000e+00	0.0000e+00	1.3544e-02	**9.8573e-03**	**9.8573e-03**
LVM1	1.2440e-02	1.0367e-02	8.2935e-03	1.5704e-32	1.5704e-32	1.5704e-32	2.0867e-32	**1.8931e-32**	2.0867e-32
LVM2	2.6235e-31	8.7899e-04	4.1449e-03	1.3497e-32	1.4730e-32	1.4730e-32	**3.8765e-32**	7.9441e-32	1.5771e-31
HTMF	8.3254e-02	8.4687e-02	1.1010e-01	3.5640e-04	1.1193e-03	1.4795e-03	7.4288e-02	**7.2117e-02**	8.5187e-02
MTYS	8.2243e-116	1.9156e-213	8.0597-274	2.5760e-127	8.5756e-232	6.2555e-295	8.2894e-120	2.6983e-220	**9.5689e-281**
MCLZ	4.3705e-02	3.5938e-02	3.3756e-02	1.8403e-02	1.8207e-02	1.8479e-02	3.5420e-02	**2.8134e-02**	2.9801e-02
NQTC	1.4412e-03	2.3647e-03	3.6301e-03	5.6536e-04	6.6833e-04	1.3726e-03	**1.4087e-03**	2.3879e-03	3.5082e-03
NCRA	3.9561e+01	3.9921e+01	4.0406e+01	1.7001e+01	1.6001e+01	2.2001e+01	3.9001+01	3.9001e+01	**3.0501e+01**
PLZ1	1.8548e-32	9.4183e-33	1.4392e-02	1.5704e-33	1.5704e-33	1.5704e-33	6.7335e-33	6.7335e-33	6.7335e-33
PLZ2	1.0987e-03	8.7899e-04	3.0765e-03	1.5962e-32	1.4730e-32	1.5962e-32	**6.7115e-32**	7.2662e-32	1.4169e-31
PRDC	9.14000e-01	9.2000e-01	9.0800e-01	9.0000e-01	9.0000e-01	9.0000e-01	9.0000e-01	9.0000e-01	9.0000e-01
RAS1	3.1495e+01	3.1396e+01	3.2390e+01	1.1930e+01	1.3918e+01	2.0877e+01	3.0819e+01	**2.9825e+01**	3.0819e+01
RAS2	3.3948e+01	3.2814e+01	3.4306e+01	1.4924e+01	1.4924e+01	1.4924e+01	3.3331e+01	**3.1839e+01**	3.2834e+01
ROSB	2.9097e+01	2.5845e+01	2.9072e+01	1.7107e+01	7.8878e+00	2.5413e+00	2.3134e+01	2.2516e+01	**2.1972e+01**

4.3 Paper 5: An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps

Paper 5 empirically investigated further the performances of two PSO variants, LDIW-PSO and RIW-PSO algorithms. Various chaotic maps were incorporated into their respective IWSs to provide chaotic features that will enable the particles move to new search regions in the search space. These investigations reveal that many of the chaotic maps improved the performance of the algorithms at a higher level than the commonly used logistic map. In terms of the number of FEs, the two PSO variants generally performed best with the intermittency chaotic map. However, considering other performance measurement, none of the chaotic maps, when used with the two variants, could enable the algorithms perform well in all test problems compared to other maps.

Based on the experimental findings, it is clear that though the logistic map could make LDIW-PSO and RIW-PSO have good performances, there exist chaotic maps that can make them perform even better in terms of convergence speed, accuracy, stability and global search ability. The findings in this paper provide some useful information regarding the usage of these maps in the inertia weight strategies of PSO variants especially LDIW-PSO and RIW-PSO.

INCLUDED ARTICLES

PAPER 3

ON ADAPTIVE CHAOTIC INERTIA WEIGHTS IN PARTICLE SWARM OPTIMIZATION [9]

M. A. ARASOMWAN AND A. O. ADEWUMI

Proceedings of the 4th IEEE Symposium Series on Computational Intelligence (SSCI '13), pp. 72-79, Singapore, 2013

Doi: 10.1109/SIS.2013.6615161

PAPER 4

AN IMPROVED PARTICLE SWARM OPTIMIZER BASED ON SWARM SUCCESS RATE FOR GLOBAL OPTIMIZATION PROBLEMS

M. A. ARASOMWAN AND A. O. ADEWUMI

Submitted to Journal of Experimental & Theoretical Artificial Intelligence.

Manuscript Number: TETA-2013-0192

PAPER 5

AN INVESTIGATION INTO THE PERFORMANCE OF PARTICLE SWARM OPTIMIZATION WITH VARIOUS CHAOTIC MAPS [12]

M. A. ARASOMWAN AND A. O. ADEWUMI

Mathematical Problems in Engineering, vol. 2014, Article ID 178959, 17 pages, 2014. doi:10.1155/2014/178959

Chapter 5

Simplified Particle Swarm Optimization

5.1. Paper 6: On the Performance of Particle Swarm Optimization with(out) some Control Parameters for Global Optimization

The efficient optimizing power of the PSO algorithm lies in the balancing of exploration and exploitation activities. Meanwhile, inertia weight, acceleration constants, random factors and velocity threshold play important roles in the exploration and exploitation ability of the PSO algorithm. Their selections could be problem-dependent, labour intensive and time consuming with the exception of random factors. Several PSO variants depend on these parameters.

Two major goals were achieved in Paper 6. Firstly, the paper experimentally demonstrated that the basic PSO (BPSO) technique can perform efficiently without using some (or any) of the control parameters in the particle velocity update formula. Secondly, the problem of premature convergence associated with the PSO technique when optimizing high dimensional multi-modal optimization problems was addressed. In achieving these goals, some modifications were made to the BPSO to make it simpler but more effective without additional complex computational efforts, to form another PSO variant branded as the modified BPSO (M-BPSO).

The modifications were inspired by the drawbacks of the BPSO with respect to premature convergence, weak local search ability and the desire to make the algorithm simpler but more efficient. Some of the modifications involved making the velocity limits of the particles decrease dynamically depending on the progressive minimum and maximum dimensional values of the entire swarm. The decreasing nature of the velocity limits was used to control the exploration and exploitation activities of M-BPSO.

Results obtained from the numerical simulations confirm that the inertia weight parameter may not always be necessary for PSO algorithms to work effectively. Also, it was discovered from the experiments that with proper modifications to some other parts of PSO algorithms, the acceleration coefficients and random factors may not be necessary in the particle velocity updating equation to obtain global optimal solutions to optimization problems.

INCLUDED ARTICLES

PAPER 6

ON THE PERFORMANCE OF PARTICLE SWARM OPTIMIZATION WITH(OUT) SOME CONTROL PARAMETERS FOR GLOBAL OPTIMIZATION

M. A. ARASOMWAN AND A. O. ADEWUMI

Submitted to the International Journal of Bio-Inspired Computation

Manuscript ID: IJBIC_73503

Chapter 6

Particle Swarm Optimization Hybrid with Local Search

6.1. Paper 7: Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems

Naturally, the PSO technique combines local search idea (through self-experience) with global search method (through neighbouring experience), in an attempt to balance exploration and exploitation activities. However, it is widely accepted that the PSO technique has good global search ability but weak local search ability because it can easily locate areas in the solution space where good solutions can be discovered. However, finding the best solution is a challenge. This difficulty often traps the PSO in local optimum leading to premature convergence.

The optimizing strategy of the PSO hinges on the sharing of new discoveries by each particle in the swarm with neighbours, while the particle with the best discovery attracts others. Though, this strategy seems to be very promising, there is the risk that the particles would be susceptible to premature convergence, especially when the problem to be optimized is multi-modal and has high dimensionality. This is due to the fact that, the more particles share their discoveries among themselves, the more likely they are to have identical behaviours, until they converge to the same area in the solution search space. If none of the particles could discover the global best, then, at some point, all the particles will converge about the existing global best and this may in turn not be the global minimizer.

For the PSO technique to leave up to expectation, it must possess a major feature that characterizes an efficient optimization algorithm, which is the ability to strike a balance between local and global searches. As a result, one of the possible ways to prevent this premature convergence is to embed a local search technique into the PSO

algorithm to help improve the quality of each solution by searching its neighbourhood. Once this is accomplished, better information is communicated among the particles thereby increasing the algorithm's ability to locate better global solutions during the course of optimization.

Paper 7 was motivated by the possibility of premature convergence associated with the idea of other particles following the best particle among them in search for a global solution within the search space relative to the optimization problem being solved. In the paper, a different local search technique was proposed to harness the global search ability of PSO and improve on its local search efforts. The local search technique is based on the collective efforts of randomly selected (with replacement) particles, which are chosen a number of times equal to the size of the problem dimension. Each particle selected is made to contribute the value in the position of its randomly selected dimension from the personal best. The contributed values are then used to form a potential global best solution which is further refined to locate a better solution in comparison to the current global solution. Two PSO variants, LDIW-PSO and RIW-PSO, which have been claimed to be less efficient in optimizing many continuous optimization problems, were used to validate the proposed improvement of the performance of the PSO technique.

Another achievement made in the paper is the improvement of the decision making strategy by the swarm in obtaining potential global solutions in the search space.

INCLUDED ARTICLES

PAPER 7

IMPROVED PARTICLE SWARM OPTIMIZATION WITH A COLLECTIVE LOCAL UNIMODAL SEARCH FOR CONTINUOUS OPTIMIZATION PROBLEMS [13]

M. A. ARASOMWAN AND A. O. ADEWUMI

Special Issue on Bioinspired Computation and Its Applications in Operation Management (BIC), *The Scientific World Journal*, vol. 2014, Article ID 798129, 23 pages, 2014. doi:10.1155/2014/798129
Manuscript ID: 798129

Chapter 7

Solving High Dimensional Problems with Particle Swarm Optimization

7.1 Paper 8: An Adaptive Velocity Particle Swarm Optimization for High-Dimensional Function Optimization

PSO variants that have been used to solve optimization problems up to 2,000 dimensions without losing superiority to its competitor(s) are not common in literature. There are possibilities of encountering high-dimensional real-world optimization problems. Therefore, PSO algorithm needs to be improved to enhance it for better performance in handling such problems.

Presented in Paper 8 is a simple PSO variant, Adaptive Velocity PSO (AV-PSO) which adaptively adjusts the velocity of particles based on Euclidean distance between the position of each particle and the position of the global best particle. The variant was implemented without using the inertia weight, acceleration coefficients and random coefficients parameters in the velocity formula for particle in the swarm. A chaotic feature was introduced into the particle's position formula to promote some stochasticity in order to facilitate good exploitation. Numerical simulations were conducted to compare the performance of AV-PSO with Adaptive Inertial weight PSO (AIWPSO), Rank based PSO (PSO_{rank}), Chaotic Random Inertia Weight PSO (CRIW-PSO), Decreasing exponential function PSO (def-PSO), Natural Exponential inertia weight PSO (e_1-PSO) and Adaptive PSO (APSO). It was also compared with Line Search Restart (LSRS) optimization technique.

Continuous optimization problems with low (10 – 30) and high (50 – 4,000) dimensions were used in the experiments. In all the experiments AV-PSO outperformed all its competitors showing that PSO is very much suitable for large-scale global optimization problems involving very high dimensions, with very good

performance in locating quality global optimal solutions with few numbers of iterations without easily getting stuck in local optimal.

INCLUDED ARTICLES

PAPER 8

AN ADAPTIVE VELOCITY PARTICLE SWARM OPTIMIZATION FOR HIGH-DIMENSIONAL FUNCTION OPTIMIZATION [10]

M. A. ARASOMWAN AND A. O. ADEWUMI

Proceedings of the IEEE Congress on Evolutionary Computation (CEC), 2013.
Mexico City, June 20-23 pp. 2352 - 2359

DOI: http://dx.doi.org/10.1109/CEC.2013.6557850

Chapter 8

Conclusion, Summary and Future Research

Motivated by the drawbacks of the PSO technique viz-a-viz premature convergence, weak local search ability as well as the desire to make the technique simpler, more effective, efficient and robust than existing variants in handling both simple and complex optimization problems, a series of studies were conducted with promising results that are reported in this thesis.

8.1 Conclusions

A number of new PSO variants are proposed in this thesis that effectively addressed the drawbacks of PSO technique namely, premature convergence and weak local search ability. Efforts were made to make the technique simpler and more effective, efficient and robust in handling problems with many local optima. Results obtained from these variants were compared among themselves and with available ones in literature in order to show their superiority.

The variants introduced in this thesis tried to avoid the introduction of additional parameters, complexities or more computational efforts unlike several other PSO variants in literature. We also introduced some dynamics into the control of the particle velocity limits and search space limits during execution of PSO as opposed to many other existing variants. Moreover, the pure greedy method of obtaining the swarm global best among the personal bests of all the particles in the swarm which is a common attribute of very many of the existing PSO variants was complemented with random and adaptive features. Some of the variants were implemented without the inertia weight, acceleration constants, random factors and the cognitive component of the velocity formula. All these clearly provide expected answers to the research questions raised in Chapter 1.

The new variants were validated with several test problems of diverse complexities and dimensionality. Experimental results obtained show substantial evidences that the variants are much better than their original counterparts and many variants in the literature in terms of reliability, robustness, convergence speed, solution quality, search ability and efficiency. Since the trends in global optimization focus nowadays on the application of metaheuristics to practical problems arising from the industries [1, 2, 23-25, 85], this thesis offers researchers with efficient variants of PSO that we believe will be of great help in solving industrial problems. These variants offer alternatives to many currently available algorithms for solving global optimization problems in which the gradient information is not readily available. They are available for optimization researchers and the results can also serve as benchmark on which further research could be based.

8.2 Summary of contributions

We provide a summary of the contributions made through the series of studies carried out as highlight over this thesis below:

- **i.** New variants of PSO which use swarm success rate as feedback parameter into their inertia weight strategies are proposed to enhance the explorative and exploitative power of the PSO technique.
- **ii.** The basic PSO was modified to propose another variant with seven versions, which use dynamic velocity limits instead of inertia weight to control its global and local search activities.
- **iii.** This work also introduced a new improved PSO with dynamic search space and velocity limits.
- **iv.** Another novel variant was proposed which diversified the operations of PSO by incorporating randomness and adaptivity to complement the greedy method PSO normally use to choose the global best among the personal bests of particles among the swarm.
- **v.** A new local search technique was proposed to address the weak local search ability of PSO technique. Promising results from this local search technique show that it can also be used with any other population-based optimization

algorithms to obtain quality solutions to simple and complex optimization problem.

vi. The results obtained from the experiments with various chaotic maps provide a platform for informative decision making by practitioners in the process of selecting chaotic maps to be used in the inertia weight formula of LDIW-PSO and RIW-PSO.

8.3 Future research

Despite the depth of experimental study conducted in this thesis, there is still room for improvement and future study. Two major areas stand out clearly for future research study namely,

i. The application of the proposed variants to real-world problems with diverse complexities especially combinatorial optimization problems and adaptation of the variants to handling constrained global optimization problems.

ii. Further study on the tuning of the parameters that makes up the proposed local search technique.

iii. Study on the parameters and behaviour of other SI techniques especially more recent ones, in search of improved techniques that can handle increasingly complex real-world optimization problems.

iv. Finally, since PSO exhibits an implicit parallelism as a multi-agent based technique, it would be worthwhile to explore a multi-agent-based framework (see [19, 97]) in the implementation of the variants which perhaps might further improve their efficiency and convergence.

References

[1] Adewumi, A.O., Sawyerr, B.A, Ali, M.M. (2009). A heuristic solution to the university timetabling problem, *Engineering Computations*, Emerald Group, Vol. 26, Issue 8, pp. 972-984.

[2] Adewumi, A.O. and Ali, M.M. (2010). A multi-level genetic algorithm for a multi-stage space allocation problem. Mathematical and Computer Modelling, Vol 51, no. 1, pp. 109-126.

[3] Afaq, H. and Saini, S. (2011), On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques, *International Journal of Computer Science Issues*, Vol. 8, Issue 4, No. 2, pp. 326-334.

[4] Akbari, R., and Ziarati, K. (2011), A rank based particle swarm optimization algorithm with dynamic adaptation, *Journal of Computational and Applied Mathematics*, Elsevier, Vol. 235, pp. 2694–2714.

[5] Alfi, A. (2011), PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems, *ACTA, Automatic Sinica*, Vol. 37, No. 5, pp. 541-549.

[6] Al-Hassan, W., Fayek, M. B. and Shaheen, S. I. (2006), PSOSA: An optimized particle swarm technique for solving the urban planning problem, *The 2006 International Conference on Computer Engineering and Systems*, pp. 401-405.

[7] Ali, M. M. and Kaelo, P. (2008), Improved particle swarm algorithms for global optimization, *Applied Mathematics and Computation*, Vol. 196, Iss. 2, pp. 578-593.

[8] Ali, M.M, Khompatraporn, C. and Zabinsky, Z.B. (2005). A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems. Journal of Global Optimization, Vol., Issue 4, pp 635-672

[9] Arasomwan, M.A. and Adewumi, A.O. (2013). On Adaptive Chaotic Inertia Weight for Particle Swarm Optimization. IEEE Symposium on Swarm Intelligence (SIS), In the Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI 2013), SS-0392, April 15-19, 2013, Singapore.

[10] Arasomwan, M.A. and Adewumi, A.O. (2013). An Adaptive Velocity Particle Swarm Optimization for High-Dimensional Function Optimization. Proceedings of the Congress on Evolutionary Computation (CEC 2013) Conference, June 20-23, Mexico (To appear).

[11] Arasomwan, M.A. and Adewumi, A.O. (2013). On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization. *The Scientific World Journal,* vol. 2013, Article ID 860289, 12 pages, 2013. doi:10.1155/2013/860289.

[12] Arasomwan, M.A. and Adewumi, A.O. (2014). An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps. *Mathematical Problems in Engineering,* vol. 2014, Article ID 178959, 17 pages, 2014. doi:10.1155/2014/178959

[13] Arasomwan, M.A. and Adewumi, A.O. (2014). Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems. Special Issue on Bioinspired Computation and Its Applications in Operation Management (BIC), *The Scientific World Journal*, vol. 2014, Article ID 798129, 23 pages, 2014. doi:10.1155/2014/798129

[14] Bakwad, K. M., Pattnaik, S. S., Sohi, B. S., Devi, S., Panigrahi, K. B., Das, S., et al.(2009), Hybrid bacterial foraging with parameter free PSO, *World Congress on Nature & Biologically Inspired Computing*, pp. 1077-1081.

[15] Barrera, J. and Flores, J. J. (2008), Equivalence of the Constriction Factor and Inertia Weight Models in Particle Swarm Optimization: A Geometric Series Analysis, *Seventh Mexican International Conference on Artificial Intelligence*, pp. 188-191.

[16] Bastos-Filho, C. J. A., Oliveira, M. A. C., Nascimento, D. N. O. and Ramos, A. D. (2010), Impact of the Random Number generator quality on particle swarm optimization algorithm running on graphic processor units, *10th International Conference on Hybrid Intelligent Systems (HIS)*, Atlanta GA, 23-25 Aug. pp. 85-90.

[17] Bishop, J. M. (1989), Stochastic Searching Networks, Proceedings 1st IEEE Int. Conf. on Artificial Neural Networks, pp. 329-331, London, UK.

[18] Bishop, J. (1989), Anarchic Techniques for Pattern Classification, PhD Thesis, University of Reading.

[19] Blamah, N.V, Adewumi, A.O. Wajiga, G.M. and Baha, B.Y. An Intelligent Particle Swarm Optimization Model based on Multi. Agent System. *African Journal of Computing & ICTs.* Vol 6, no. 3. Pp1-8.

[20] Bratton, D. and Kennedy, J. (2007), Defining a Standard for Particle Swarm Optimization, *IEEE Swarm Intelligence Symposium, SIS 2007*, pp. 120-127.

[21] Cai, X., Cui, Z., Zeng, J. and Tan, Y. (2009), Individual parameter selection strategy for particle swarm optimization, *Particle Swarm Optimization*, Aleksandar Lazinica (Ed.), InTech, pp. 89-112

[22] Chetty, S. and Adewumi, A.O. (2013). Three new stochastic local search algorithms for continuous optimization problems. *Computational Optimization and Applications Journal*. Springer Publishing, vol. 56, no. 3, pp. 675–721.

[23] Chetty, S. and Adewumi, A.O. (2013). Three New Stochastic Local Search Metaheuristics for the Annual Crop Planning Problem based on a New Irrigation Scheme. *Journal of Applied Mathematics*, vol. 2013, Article ID 158538, 14 pages, 2013. doi:10.1155/2013/158538

[24] Chetty, S. and Adewumi, A.O. (2013). Comparison of Swarm Intelligence Meta-heuristics for the Annual Crop Planning Problem. Accepted for *IEEE Transactions on Evolutionary computations*. DOI: 10.1109/TEVC.2013.2256427. In Press.

[25] Chetty, S. and Adewumi, A.O. (2013). Studies in swarm intelligence techniques for annual crop planning problem in a new irrigation scheme. *South African Journal of Industrial Engineering,* vol. 24, no. 3, pp. 205–226, 2013.

[26] Chu, S.-C., Tsai, P.-W. and Pan, J.-S. (2006), Cat Swarm Optimization, *PRICAI 2006: Trends in Artificial Intelligence*, Vol. 4099, Q. Yang and G. Webb, Eds., ed: Springer Berlin Heidelberg, pp. 854-858.

[27] Chuang, L-Y., Yang, C-H., and Li, J-C. (2011). Chaotic maps based on binary particle swarm optimization for feature selection, *Applied Soft Computing* Vol. 11, pp. 239–248.

[28] Chauhan, P., Deep, K. and Pant, M. (2013), Novel inertia weight strategies for particle swarm optimization, *Memetic Computing*, Vol. 5, Issue. 3, pp. 229–251.

[29] Chen, G. H., Jia, X. J. and Min, Z. (2006), Natural exponential Inertia Weight strategy in particle swarm optimization, *Sixth World Congress on Intelligent Control and Automation, WCICA*, June 21 - 23, Dalian, China, Vol. 1, pp. 3672–3675.

[30] Clerc, M. and Kennedy, J. (2002), "The particle swarm - explosion, stability, and convergence in multidimensional complex space", *IEEE Transactions on Evolutionary Computation,* Vol. 6, pp. 58-73.

[31] Coelho, L. D. S. (2008), A quantum particle swarm optimizer with chaotic mutation operator, *Chaos, Solutions and Fractals*, Elsevier, Vol. 37, pp. 1409-1418.

[32] Das, S., Biswas, A., Dasgupta, S. and Abraham, A. (2009), Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, in *Foundations of Computational Intelligence Volume 3*. vol. 203, A. Abraham, A.-E. Hassanien, P. Siarry, and A. Engelbrecht, (Eds.), Springer Berlin Heidelberg, pp. 23-55.

[33] De Castro, L. N. and Timmis, J. (2002), *Artificial Immune Systems: A New Computational Intelligence Approach*, Springer, pp. 57–58.

[34] Dorigo, M. (1992), *Optimization, Learning and Natural Algorithms*, PhD thesis, Politecnico di Milano, Italie.

[35] Dorigo, M. and Birattari, M. (2007), Swarm intelligence, *Scholarpedia*, Vol. 2, Issue. 9, pp. 1462.

[36] Eberhart, R. C. and Kennedy, J. (1995), A new optimizer using particle swarm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95*, Nagoya, Japan, pp. 39-43.

[37] Eberhart, R. C. and Shi, Y. (2002), Tracking and optimizing dynamic systems with particle swarms, *Proceedings of the 2001 Congress on Evolutionary Computation*, Vol. 1, pp. 94–100.

[38] Eberhart, R. C. and Shi, Y. (2000), Comparing inertia weights and constriction factors in particle swarm optimization, *Proceedings of the 2000 Congress on Evolutionary Computation*, Vol. 1, pp. 84-88

[39] Ememipour, J., Nejad, M. M. S., Ebadzadeh, M. M., and Rezanejad, J. (2009), Introduce a new inertia weight for particle swarm optimization, *The Fourth International Conference on Computer Sciences and Convergence Information Technology*, pp. 1650-1653.

[40] Evers, G. I. (2009). *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization*. MSc. Thesis. University of Texas-Pan American.

[41] Feng, Y., Teng, G.F., Wang, A.X., and Yao, Y.M., (2008), Chaotic Inertia Weight in Particle Swarm Optimization, *Second International Conference on Innovative Computing, Information and Control*, pp. 475.

[42] Feng, X., Lau, F. C. M. and Yu, H. (2013), A novel bio-inspired approach based on the behavior of mosquitoes, *Inf. Sci.,* vol. 233, pp. 87-108.

[43] Feng, C., Xinxin, S. and Dali, W. (2011), Inertia Weight Particle Swarm Optimization with Boltzmann Exploration, *Seventh International Conference on Computational Intelligence and Security (CIS)*, pp. 90-95.

[44] Gabere, M. N. (2007), Simulated Annealing Driven Pattern Search Algorithms for Global Optimization, MSc. Thesis, School of Computational and Applied Mathematics, University of the Witwatersrand, South Africa.

[45] Gandomi, A. H., Yun, G. J., Yang, X-S., Talatahari, S. (2013), Chaos-enhanced accelerated particle swarm optimization, *Commu Nonlinear Sci Numer Simulat*. Vol. 18, pp. 327-340.

[46] Gao, Y., An, X. and Liu, J. (2008), A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation, *International Conference on Computational Intelligence and Security*, Vol. 1, pp. 61–65.

[47] Gao, Y.-L. and Duan, Y.-H. (2007), A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy, *International Conference on Computational Intelligence and Security Workshops, CISW 2007*, pp. 199-203.

[48] Garnier, S., Gautrais, J. and Theraulaz, G. (2007), The biological principles of swarm intelligence, *Swarm Intell*, Vol. 1, pp. 3–31.

[49] Glover, F. (1986), Future paths for integer programming and links to artificial intelligence, *Computers and Operations Research*, Vol. 13, No. 5, pp. 533-549.

[50] Guo, W. Chen, G. and Feng, X. (2006), A new strategy of acceleration coefficients for particle swarm optimization, *Proceedings of the 10th IEEE International Conference on Computer Supported Cooperative Work in Design*, pp. 1-5.

[51] Hao, P., Ming, Z. and Xiaolei, H. (2008), Particle swarm-simulated annealing fusion algorithm and its application in function optimization, *2008 International Conference on Computer Science and Software Engineering*, pp. 78-81.

[52] Havens, T. C., Spain, C. J., Salmon, N. G. and Keller, J. M. (2008), Roach Infestation Optimization, *IEEE Swarm Intelligence Symposium, SIS 2008*, St. Louis, MO, 21-23 Sept., pp. 1-7.

[53] Hu, M., Wu, T. and Weir, J. D. (2013), An Adaptive Particle Swarm Optimization With Multiple Adaptive Methods, *IEEE Transactions on Evolutionary Computation*, Vol. 17, No. 5, pp. 705-720.

[54] Jiang, Y., Hu, T., Huang, C. and Wu, X. (2007), An improved particle swarm optimization algorithm, *Applied Mathematics and Computation*, Vol. 193, Iss. 1, pp. 231-239.

[55] Jiao, B., Lian, Z. and Gu, X. (2006), A dynamic inertia weight particle swarm optimization algorithm, *Chaos, solutions and fractals*, Elsevier, No. 37, pp. 698-705.

[56] Jing, B., Xueying, Z. and Yueling, G. (2009), Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system, in *Mechatronics and Automation, ICMA 2009, International Conference on*, pp. 4754-4759.

[57] Jordehi, A. R. and Jasni, J. (2013), Parameter selection in particle swarm optimisation: a survey, *Journal of Experimental & Theoretical Artificial Intelligence*, Vol. 25, No. 4, pp. 527-542

[58] Jun, T. and Xiaojuan, Z. (2009), Particle swarm optimization using adaptive local search, *International Conference on Future BioMedical Information Engineering*, pp. 300-303.

[59] Junying, C., Zheng, Q., Yu, L. and Jiang, L. (2005), Particle swarm optimization with local search, *International Conference on Neural Networks and Brain, ICNN&B '05.*, 2005, pp. 481-484.

[60] Karaboga, D. (2005), An Idea Based On Honey Bee Swarm for Numerical Optimization, Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

[61] Kaveh, A. and Talatahari, S. (2010), A novel heuristic optimization method: charged system search, *Acta Mechanica,* Vol. 213, pp. 267-289.

[62] Kaveh, A. and Talatahari, S. (2011), An enhanced charged system search for configuration optimization using the concept of fields of forces, *Struct Multidiscip Optim*, Vol. 43, No. 3, pp. 339-351.

[63] Kennedy, J. and Eberhart, R. C. (1995), Particle swarm optimization, In Proceedings of IEEE international conference on neural networks, Vol. 4, pp. 1942–1948. Perth, Australia.

[64] Kentzoglanakis, K. and Poole, M. (2009), Particle swarm optimization with an oscillating Inertia Weight, *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pp. 1749–1750.

[65] Kephart, J. O. (1994), A biologically inspired immune system for computers *Proceedings of Artificial Life IV: The Fourth International Workshop on the Synthesis and Simulation of Living Systems*, MIT Press, pp. 130–139.

[66] Krishnanand, K. N. and Ghose, D. (2006), Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications, *Multi-agent and Grid Systems*, Vol. 2, No. 3, pp. 209–222.

[67] Lei, W. and Zhaocai, X. (2008), The research of PSO algorithms with non-linear time-decreasing inertia weight, *7th World Congress on Intelligent Control and Automation*, pp. 4002-4005.

[68] Li, H. R., and Gao, Y. L. (2009), Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation, *Second International Conference on Information and Computing Science*, pp. 66–69.

[69] Lili-Li and Xingshi-He. (2009), Gaussian mutation Particle Swarm Optimization with dynamic adaptation inertia weight, In World Congress on Software Engineering, IEEE, pp. 454-459.

[70] Liu, B., Wang, L., Jin, Y-H., Tang, F. and Huang D-X. (2005), Improved particle swarm optimization combined with chaos, *Chaos, Solutions and fractals*, Vol. 25, pp. 1261-1271.

[71] Mai, X.-F. and Li, L. (2012), Bacterial foraging algorithm based on gradient particle swarm optimization algorithm, *Eighth International Conference on Natural Computation (ICNC)*, pp. 1026-1030.

[72] Malik, R. F., Rahman, T. A., Hashim, S. Z. M. and Ngah, R. (2007), New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight, *International Journal of Computer Science and Security (IJCSS)*, Vol. 1, No. 2, pp. 35.

[73] Mansour, M. M., Mekhamer, S. F. and El-Kharbawe, N. E. S. (2007), A Modified Particle Swarm Optimizer for the Coordination of Directional Overcurrent Relays, *Power Delivery, IEEE Transactions,* Vol. 22, pp. 1400-1410.

[74] Meyer, K. D, (2004), *Foundations of Stochastic Diffusion Search*, PhD Thesis, The University of Reading.

[75] Monismith, D. R. and Mayfield, B. E. (2008), Slime Mold as a model for numerical optimization, *IEEE Swarm Intelligence Symposium, SIS 2008*, pp. 1-8.

[76] Ngoc, D. V., Schegner, P., and Ongsakul, W. (2011), A newly improved particle swarm optimization for economic dispatch with valve point loading effects, in *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1-8.

[77] Nickabadi, A., Ebadzadeh, M. M. and Safabakhsh, R. (2011), A novel particle swarm optimization algorithm with adaptive inertia weight, *Applied soft computing*, Vol. 11, pp. 3658-3670.

[78] Niu, S. H., Ong, S. K. and Nee, A. Y. C. (2012), An improved Intelligent Water Drops algorithm for achieving optimal job-shop scheduling solutions, *International Journal of Production Research*, Vol. 50, No. 15, pp. 4192–4205.

[79] Ólafsson, S. (2006), Metaheuristics, Nelson and Henderson (eds.), Handbook on Simulation, Handbooks in Operations Research and Management Science VII, Elsevier, pp. 633-654.

[80] Palanikkumar, D., Gowsalya, E., Rithu, B. and Anbuselven, P. (2012), An intelligent water drops algorithm based service selection and composition in service oriented architecture, *Journal of Theoretical and Applied Information Technology*, Vol. 39, No.1, pp.45-51.

[81] Pant, M., Radha, T. and Singh, V. P. (2007), Particle Swarm Optimization Using Gaussian Inertia Weight, *International Conference on Computational*

Intelligence and Multimedia Applications, Sivakasi, Tamil Nadu, 13-15 Dec, Vol. 1, pp. 97-102.

[82] Parpinelli, R. and Lopes, H. S. (2012), Theory and New applications of swarm intelligence, InTech, Croatia, Edited.

[83] Passino, K. M. (2002), Biomimicry of Bacterial Foraging for Distributed Optimization and Control, *IEEE Control Systems Magazine*, pp. 52–67.

[84] Pedersen, M. E. H. (2010), Good parameters for particle swarm optimization, Technical report, Hvass laboratories, number HL1001.

[85] Phillips, R., Woolway, M., Fanucchi, D. and Ali, M.M. (2014). Mathematical Modeling and Optimal Blank Generation in Glass Manufacturing, Special issue on Mathematical Modeling and Optimization of Industrial Problems, *Journal of Applied Mathematics*, to appear in 2014.

[86] Poli, R., Kennedy, J. and Blackwell, T. (2007), Particle swarm optimization, *Swarm Intelligence,* Vol. 1, pp. 33-57.

[87] Rabanal, P., Rodríguez, I., and Rubio, F. (2007), Using river formation dynamics to design heuristic algorithms, *Unconventional Computation, UC'07*, LNCS 4618, pp. 163–177.

[88] Rabanal, P., Rodríguez, I., and Rubio, F. (2010), Applying RFD to Construct Optimal Quality-Investment Trees, *Journal of Universal Computer Science*, Vol. 16, No. 14, pp.1882-1901.

[89] Rabanal, P., Rodríguez, I. and Rubio, F. (2011), Studying the application of ant colony optimization and river formation dynamics to the Steiner tree problem, *Evol. Intel.*, Vol. 4, pp. 51–65.

[90] Saffarzadeh, V. M., Jafarzadeh, P. and Mazloom, M. (2010), A hybrid approach using particle swarm optimization and simulated annealing for n-queen problem, *World academy of science, engineering and technology*, Vol. 67, pp. 517-521.

[91] Sawyerr, B. A., Adewumi, A.O. and Ali, M.M. (2013). Benchmarking Projection-Based Real Coded Genetic Algorithms on BBOB-2012 Noiseless Function Testbed, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013), pp. 1193–1200

[92] Sawyerr, B. A., Adewumi, A.O. and Ali, M.M (2014). Real-coded genetic algorithm with uniform random local search, *Applied Mathematics and Computation*, vol. 228, 589--597, 2014.

[93] Schutte, J. F., Reinbolt, J. A., Fregly, B. J., Haftka, R. T. and George, A. D. (2004), Parallel global optimization with the particle swarm algorithm, *International Journal for numerical methods and engineering*, Vol. 61, pp. 2296–2315.

[94] Sedighizadeh, D., and Masehian, E. (2009), Particle swarm optimization methods, taxonomy and applications, *International Journal of computer theory and engineering,* Vol. 1, No. 5, pp. 486-502.

[95] Selvakumar, A. I. and Thanushkodi, K. (2007), A new particle swarm optimization solution to nonconvex economic dispatch Problems. *IEEE Trans. on Power Systems*, Vol. 22, No. 1, pp. 42-51.

[96] Selvakumar, A. I. and Thanushkodi, K. (2008), Anti-predatory particle swarm optimization: Solution to nonconvex economic dispatch problems, *Electric Power Systems Research*, Vol. 78, pp. 2-10.

[97] Shangxiong S. (2008). A Particle Swarm Optimization (PSO) Algorithm Based on Multi-agent System. Proceedings of the 2008 International Conference on Intelligent Computation Technology and Automation (ICICTA '08), Vol. 2, pp 802-805.

[98] Shah-Hosseini, H. (2007), Problem solving by intelligent water drops, *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2007*, Singapore, pp. 3226–3231.

[99] Shah-Hosseini, H., (2009), The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm, *International Journal of Bio-Inspired Computation*, Vol. 1 (1–2), pp.71–79.

[100] Shah-Hosseini, H., (2008), Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem, *International Journal of Intelligent Computing and Cybernetics*, Vol. 1, No. 2, pp. 193 – 212.

[101] Shen, X., Chi, Z., Yang, J., Chen, C. and Chi, Z. (2010), Particle swarm optimization with dynamic adaptive inertia weight. *IEEE International Conference on Challenges in Environmental Science and Computer Engineering*, pp. 287-290.

[102] Shen, X., Chi, Z., Yang, J., Chen, C. and Chi, Z. (2010), Particle swarm optimization with dynamic adaptive inertia weight, *IEEE International Conference on Challenges in Environmental Science and Computer Engineering*, pp. 287-290.

[103] Shi, Y. and Eberhart, R. (1998), Parameter selection in particle swarm optimization, in *Evolutionary Programming VII*. V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., ed: Springer Berlin Heidelberg, Vol. 1447, pp. 591-600.

[104] Shi, Y. and Eberhart, R. C. (1998), A modified particle swarm optimizer, *Proceedings of the IEEE international conference on evolutionary computation*, pp. 69–73.

[105] Shi, Y. H. and Eberhart, R. C. (2001), Fuzzy adaptive particle swarm optimization, *Congress on evolutionary computation*, Korea.

[106] Shu, J. and Li, J. (2009), An improved self-adaptive particle swarm optimization algorithm with simulated annealing, *Third International Symposium on Intelligent Information Technology Application*, pp. 396-399.

[107] Su, T-J., Cheng, J-C. and Sun, Y-D. (2011), Particle swarm optimization with time-varying acceleration coefficients based on cellular neural network for colour image noise cancellation, *Sixth international conference on digital telecommunications*, ICDT, pp. 109-115.

[108] Sun. J., Lai, C-H., and Wu, X-J. (2011), Particle swarm optimization: classical and quantum perspectives, CRC press, New York.

[109] Talatahari, S., Kaveh, A. and Sheikholeslami, R. (2011), An efficient charged system search using chaos for global optimization problems, *International Journal of Optimization in Civil Engineering*, Vol. 2, pp. 305-325.

[110] Tavazoei, M. S. and Haeri, M. (2007), Comparison of different one-dimension maps as chaotic search pattern in chaos optimization algorithms, *Applied Mathematics and Computation*, Vol. 187, pp.1076-1085.

[111] Thangaraj, R., Pant, M., Abraham, A. and Bouvry, P. (2011), Particle swarm optimization: Hybridization perspectives and experimental illustrations, *Applied Mathematics and Computation*, Vol. 217, No. 12, pp. 5208-5226.

[112] Timmis, J., Neal, M. and Hunt, J. (2000), An artificial immune system for data analysis, *BioSystems*, Vol. 55**,** No. 1, pp. 143–150.

[113] Trelea, C. I. (2003), The particle swarm optimization algorithm: convergence analysis and parameter selection, *Information Processing Letter*, Vol. 85, pp. 317-325.

[114] Wei-Bo, Z., Jin-You, C. and Ya-Qiang, Y. (2010), Study on particle swarm optimization algorithm with local interpolation search, *2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR)*, pp. 345-348.

[115] Waibel, M., Floreano, D. and Keller, L. (2011), A quantitative test of Hamilton's rule for the evolution of altruism, *PLoS Biology*, Vol. 9, No. 5, pp. 1-7.

[116] Wenhua, H., Ping, Y., Haixia, R. and Jianpeng, S. (2010), Comparison study of several kinds of inertia weights for PSO, *IEEE International Conference on Progress in Informatics and Computing (PIC)*, pp. 280-284.

[117] Xiaohui, H., Yuhui, S. and Eberhart, R. (2004), Recent advances in particle swarm, in *Congress on Evolutionary Computation*, Vol. 1, pp. 90-97.

[118] Xin, J., Chen, G. and Hai, Y. (2009), A Particle Swarm Optimizer with Multi-Stage Linearly-Decreasing Inertia Weight, *International Joint Conference on Computational Sciences and Optimization*.

[119] Xin-She, Y. and Deb, S. (2009), Cuckoo Search via Lévy flights, *World Congress on Nature & Biologically Inspired Computing, NaBIC 2009*, pp. 210-214.

[120] Xin-She, Y. (2012). Swarm-Based Metaheuristic Algorithms and No-Free-Lunch Theorems, *Theory and New applications of swarm intelligence*, Parpinelli, R. and Lopes, H. S. (Eds.), InTech, Croatia.

[121] Xin-She, Y. (2011), Metaheuristic Optimization, *Scholarpedia*, Vol. 6, No, 8, pp. 11472.

[122] Yan, C-M., Guo, B-L. and Wu, X-X. (2012), Empirical study of the inertia weight particle swarm optimization with constraint factor, *International Journal of Soft Computing and Software Engineering*, Vol. 2, No. 2, pp. 1-8.

[123] Yang, X. S. (2008), *Nature-Inspired Metaheuristic Algorithms*, Frome: Luniver Press.

[124] Yang, X. S. (2009), Firefly algorithms for multimodal optimization, *Stochastic Algorithms: Foundations and Applications, SAGA 2009*, Lecture Notes in Computer Science 5792, pp. 169–178.

[125] Yang, X-S. (2012), Nature-Inspired Mateheuristic Algorithms: Success and New Challenges, *Journal of Computer Engineering & Information Technology*, Vol. 1, No. 1.

[126] Yang, X. S. (2010), A New Metaheuristic Bat-Inspired Algorithm, *Nature Inspired Cooperative Strategies for Optimization* (NISCO 2010) (Eds. J. R. Gonzalez et al.), Studies in Computational Intelligence, Springer Berlin, 284, Springer, pp. 65-74.

[127] Yanxia, S., Wyk, B. J. V. and Zenghui, W. (2012), A new golden ratio local search based particle swarm optimization, *International Conference on Systems and Informatics (ICSAI)*, pp. 754-757.

[128] Yong-ling, Z., Long-hua, M., Li-yan, Z. and Ji-xin, Q. (2003), Empirical study of particle swarm optimizer with an increasing inertia weight, *The 2003 Congress on Evolutionary Computation, CEC '03*, Vol. 1, pp. 221-226.

[129] Yong-ling, Z., Long-Hua, M., Li-yan, Z. and Ji-xin, Q. (2003), On the convergence analysis and parameter selection in particle swarm optimization, *International Conference on Machine Learning and Cybernetics*, Vol. 3, pp. 1802-1807.

[130] Zhao, S. Z., Liang, J. J., Suganthan, P. N. and Tasgetiren, M. F. (2008), Dynamic multi-swarm particle swarm optimizer with local search for Large

Scale Global Optimization, in *IEEE Congress on Evolutionary Computation, (IEEE World Congress on Computational Intelligence)*, pp. 3845-3852.

[131] Zhan, Z-H., Zhang, J., Li, Y. and Chung, H.S.-H. (2009), Adaptive Particle Swarm Optimization, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 39, No. 6, pp. 1362-1381.

[132] Zhang, L-P., Yu, H-J. and Hu, S-X. (2005), Optimal choice of parameters for particle swarm optimization, *Journal of Zhejiang University SCIENCE*, Vol. 6A, No. 6, pp. 528-534.

[133] Zhang, L., Yu, H. and Hu, S. (2003), A New Approach to Improve Particle Swarm Optimization, *Genetic and Evolutionary Computation — GECCO*, Cantú-Paz, E. J. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly*, et al.*, (Eds.), Springer Berlin Heidelberg, Vol. 2723, pp. 134-139.

Appendix

Test Problems [8]

***ACKL*: Ackley Function**

Model: $ACKL(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$
Feature: Multimodal and Non-separable
Search space: $-32 \leq x_i \leq 32, i = 1, 2, \cdots, d$
Global minimum: $x^* = (0, \cdots, 0); ACKL(x^*) = 0$

***ALFP*: Aluffi Pentini's Function**

Model: $0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$
Feature: Multimodal and Separable
Search space: $-10 \leq x_i \leq 10, i = 1, 2$
Global minimum: $x^* = (-1.0465, 0); ALFP(x^*) \approx -0.3523$

***BEAL*: Beale Function**

Model: $BEAL(\vec{x}) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$
Feature: Unimodal and Non-separable
Search space: $-4.5 \leq x_i \leq 4.5, i = 1, 2$
Global minimum: $x^* = (3, 0.5); BEAL(x^*) = 0$

***BELA*: Becker & Lago Function**

Model: $BElA(\vec{x}) = (|x_1| - 5)^2 + (|x_2| - 5)^2$
Feature: Multimodal and Separable
Search space: $-10 \leq x_i \leq 10, i = 1, 2$
Global minimum: $x^* = (\pm 5, \pm 5); BELA(x^*) = 0$

***BKY1*: Bohachevsky-1 Function**

Model: $BKY1(\vec{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$
Feature: Multimodal and Separable
Search space: $-50 \leq x_i \leq 50, i = 1, 2$
Global minimum: $x^* = (0, 0); BKY1(x^*) = 0$

***BKY2*: Bohachevsky-2 Function**

Model: $BKY2(\vec{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$
Feature: Multimodal and Non-separable
Search space: $-50 \leq x_i \leq 50, i = 1, 2$

Global minimum: $x^* = (0,0); BKY2(x^*) = 0$

***BOOT*: Booth Function**

Model: $BOOT(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
Feature: Multimodal and Non-separable
Search space: $-10 \leq x_i \leq 10, i = 1,2$
Global minimum: $x^* = (1,3); BOOT(x^*) = 0$

***BRWN*: Brown-3 Function**

Model: $BRWN(\vec{x}) = \sum_i^{d-1}\left(x_i^{2\left(x_{i+1}^2+1\right)} + x_{i+1}^{2\left(x_i^2+1\right)}\right)$
Feature: Multimodal and Non-separable
Search space: $-1 \leq x_i \leq 4, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); BRWN(x^*) = 0$

***CML3*: Camel-3 Function**

Model: $CML3(\vec{x}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$
Feature: Multimodal and Non-separable
Search space: $-5 \leq x_i \leq 5, i = 1,2$
Global minimum: $x^* = (0,\cdots,0); CML3(x^*) = 0$

***CML6*: Camel-6 Function**

Model: $CML6(\vec{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$
Feature: Multimodal and Non-separable
Search space: $-5 \leq x_i \leq 5, i = 1,2$
Global minimum:
$x^* = (0.089842, -0.712656)\ or\ x^* = (-0.089842, 0.712656); CML6(x^*) = -1.0316$

***CIGR*: Cigar Function**

Model: $CIGR(\vec{x}) = X_i^2 + 10000\sum_{i=2}^d x_i^2$
Feature: Unimodal and Non-separable
Search space: $-10 \leq x_i \leq 10, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); CIGR(x^*) = 0$

***CSM*: Cosine Mixture Function**

Model: $CSM(\vec{x}) = 0.1\sum_{i=1}^d \cos(5\pi x_i) - \sum_{i=1}^d x_i^2$
Feature: Unimodal and Non-separable
Search space: $-1 \leq x_i \leq 1, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); CSM(x^*) = 0.4$

***CVLE*: Colville Function**

Model: $CVLE = (\vec{x}) = 100\left(x_1^2 - x_2\right)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90\left(x_3^2 - x_4\right)^2 +$

$$10.1\left((x_2-1)^2+(x_4-1)^2\right)+19.8(x_2-1)(x_4-1)$$

Feature: Multimodal and Non-separable
Search space: $-10 \le x_i \le 10, i = 1,2,3,4$
Global minimum: $x^* = (1,1,1,1); CVLE(x^*) = 0$

***DEJ4*: De Jong's Function**

Model: $DEJ4(\vec{x}) = \sum_{i=1}^{d} i x_i^4$
Feature: Unimodal and separable
Search space: $-5.12 \le x_i \le 5.12, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); DEJ4(x^*) = 0$

***DIXP*: Dixon Price Function**

Model: $DIXP(\vec{x}) = (x_1-1)^2 + \sum_{i=2}^{d} i\left(2x_i^2 - x_{i-1}\right)^2$
Feature: Unimodal and Non-separable
Search space: $-10 \le x_i \le 10, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); DIXP(x^*) = 0$

***EXPN*: Exponential Function**

Model: $EXPN(\vec{x}) = \exp(-0.5\sum_{i=1}^{n} x_i^2)$
Feature: Unimodal and Separable
Search space: $-1 \le x_i \le 1, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); EXPN(x^*) = -1$

***ESOM*: Easom Function**

Model: $ESOM(\vec{x}) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$
Feature: Unimodal and Non-separable
Search space: $-10 \le x_i \le 10, i = 1,2$
Global minimum: $x^* = (\pi,\pi); ESOM(x^*) = -1$

***GWNK*: Griewank Function**

Model: $GWNK(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$
Feature: Multimodal and Non-separable
Search space: $-600 \le x_i \le 600, i = 1,2,\cdots,d$
Global minimum: $x^* = (0,\cdots,0); GWNK(x^*) = 0$

***LVM1*: Levy & Mantalvo-1 Function**

Model: $LVM1(\vec{x}) = \left(\frac{\pi}{n}\right)\left(10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\right)$; where: $y_i = 1 + \frac{1}{4}(x_i+1)$
Feature: Multimodal and Non-separable
Search space: $-10 \le x_i \le 10, i = 1,2,\cdots,d$
Global minimum: $x^* = (-1,-1,\cdots,-1); LVM1(x^*) = 0$

***LVM2*: Levy & Mantalvo-2 Function**

Model: $LVM2(\vec{x}) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}(x_i - 1)^2[1 + \sin^2(3\pi x_{i+1})] + (x_n - 1)^2 \times [1 + \sin^2(2\pi x_n)])$
Feature: Multimodal and Non-separable
Search space: $-5 \leq x_i \leq 5, i = 1, 2, \cdots, d$
Global minimum: $x^* = (1, 1, \cdots, 1); LVM2(x^*) = 0$

***HTMF*: Hartman-3 Function**

Model: $HTMF(\vec{x}) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$
Feature: Multimodal and Non-separable
Search space: $0 \leq x_i \leq 1, i = 1, 2, 3$
Global minimum: $x^* = (0.114, 0.556, 0.852); HTMF(x^*) = -3.86$

***MTYS*: Matyas Function**

Model: $MTYS(\vec{x}) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
Feature: Unimodal and Non-separable
Search space: $-10 \leq x_i \leq 3102, i = 1, 2$
Global minimum: $x^* = (0.114614, 0.555649, 0.852547); MTYS(x^*) = -3.86278$

***MCLZ*: Michalewicz Function**

Model: $MCLZ(\vec{x}) = -\sum_{i=1}^{d} \sin(x_i)\left[sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m} where\ m = 10$
Feature: Multimodal and Separable
Search space: $0 \leq x_i \leq \pi, i = 1, 2$
Global minimum: $x^* = (0, \cdots, 0); MCLZ(x^*) = -1.8013$

***NQTC*: Noisy Quatic Function**

Model: $NQTC(\vec{x}) = \sum_{i=1}^{d} ix_i^4 + random(0,1)$
Feature: Unimaodal and Separable
Search space: $-1.28 \leq x_i \leq 1.28, i = 1, 2, \cdots, d$
Global minimum: $x^* = (0, \cdots, 0); NQTC(x^*) = 0$

***NCRA*: Non-Continuous Rastrigin Function**

Model: $NCRA(\vec{x}) = \sum_{i=1}^{d}(y_i^2 - 10\cos(2\pi y_i) + 10);\ y_i = \begin{cases} x_i & if\ |x_i| < 0.5 \\ \frac{round(2x_i)}{2} & if\ |x_i| \geq 0.5 \end{cases}$
Feature: Multimodal and Separable
Search space: $-5.12 \leq x_i \leq 5.12, i = 1, 2, \cdots, d$
Global minimum: $x^* = (0, \cdots, 0); NCRA(x^*) = 0$

***PLZ1*: Penalized Function-1**

Model: $PLZ1(\vec{x}) = \frac{\pi}{30}(10sin^2(\pi y_1) + \sum_{i=1}^{d-1}(y_i - 1)^2(1 + 10sin^2(\pi y_{i+1})) + (y_n - 1)^2) + \sum_{i=1}^{d} u(x_i, 10, 100, 4)$; where $u(x_i, a, k, m) =$

$$\begin{cases} k(x_i-a)^m, & \text{if } x_i > a \\ 0, & \text{if} -a \le x_i \le a; \text{ and } y_i = 1+\frac{1}{4}(x_i+1) \\ k(-x_i-a)^m, & \text{if } x_i < -a \end{cases}$$

Feature: Multimodal and Non-separable
Search space: $-50 \le x_i \le 50, i = 1, 2, \cdots, d$
Global minimum: $x^* = (1, 1, \cdots, 1); PLZ1(x^*) = 0$

PLZ2: Penalized Function-2

Model: $PLZ2(\vec{x}) = 0.1\left(sin^2(3\pi x_1) + \sum_{i=1}^{d-1}(x_i-1)^2\left(1+sin^2(3\pi x_{i+1})\right) + (x_n - 1)^2\left(1+sin^2(2\pi x_n)\right)\right) + \sum_{i=1}^{d} u(x_i, 5, 100, 4)$; where

$$u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m, & \text{if } x_i > a \\ 0, & \text{if} -a \le x_i \le a \\ k(-x_i-a)^m, & \text{if } x_i < -a \end{cases}$$

Feature: Multimodal and Non-separable
Search space: $-50 \le x_i \le 50, i = 1, 2, \cdots, d$
Global minimum: $x^* = (1, 1, \cdots, 1); PLZ2(x^*) = 0$

PRDC: Periodic Function

Model: $PRDC(\vec{x}) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1\exp(-x_1^2 - x_2^2)$
Feature: Multimodal and Non-separable
Search space: $-10 \le x_i \le 10, i = 1, 2$
Global minimum: $x^* = (0, 0); PRDC(x^*) = 0.9$

RAS1: Rastrigin Function-1

Model: $RAS1(\vec{x}) = \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10)$
Feature: Multimodal and Separable
Search space: $-5.12 \le x_i \le 5.12, i = 1, 2, \cdots, d$
Global minimum: $x^* = (0, \cdots, 0); RAS1(x^*) = 0$

RAS2: Rastrigin Function-2

Model: $RAS2(\vec{x}) = 10d + \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i))$
Feature: Multimodal and Separable
Search space: $-5.12 \le x_i \le 5.12, i = 1, 2, \cdots, d$
Global minimum: $x^* = (0, \cdots, 0); RAS2(x^*) = 0$

ROSB: Rosenbrock Function

Model: $ROSB(\vec{x}) = \sum_{i=1}^{d-1}(100(x_{i+1} - x_i^2)^2) + (x_i - 1)^2$
Feature: Unimodal and Non-separable
Search space: $-30 \le x_i \le 30, i = 1, 2, \cdots, d$
Global minimum: $x^* = (1, 1, \cdots, 1); ROSB(x^*) = 0$

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2013, Article ID 860289, 12 pages
http://dx.doi.org/10.1155/2013/860289

Research Article

On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization

Martins Akugbe Arasomwan and Aderemi Oluyinka Adewumi

School of Mathematics, Statistics, and Computer Science, University of Kwazulu-Natal, Private Bag X54001, Durban 4000, South Africa

Correspondence should be addressed to Aderemi Oluyinka Adewumi; laremtj@gmail.com

Received 9 July 2013; Accepted 4 September 2013

Academic Editors: P. Melin, G. Terracina, and G. Wei

Linear decreasing inertia weight (LDIW) strategy was introduced to improve on the performance of the original particle swarm optimization (PSO). However, linear decreasing inertia weight PSO (LDIW-PSO) algorithm is known to have the shortcoming of premature convergence in solving complex (multipeak) optimization problems due to lack of enough momentum for particles to do exploitation as the algorithm approaches its terminal point. Researchers have tried to address this shortcoming by modifying LDIW-PSO or proposing new PSO variants. Some of these variants have been claimed to outperform LDIW-PSO. The major goal of this paper is to experimentally establish the fact that LDIW-PSO is very much efficient if its parameters are properly set. First, an experiment was conducted to acquire a percentage value of the search space limits to compute the particle velocity limits in LDIW-PSO based on commonly used benchmark global optimization problems. Second, using the experimentally obtained values, five well-known benchmark optimization problems were used to show the outstanding performance of LDIW-PSO over some of its competitors which have in the past claimed superiority over it. Two other recent PSO variants with different inertia weight strategies were also compared with LDIW-PSO with the latter outperforming both in the simulation experiments conducted.

1. Introduction

The idea of Particle Swarm Optimization (PSO) stems from biology where a swarm of birds coordinates itself in order to achieve a goal. When a swarm of birds looks for food, its individuals will spread in the environment and move around independently with some degree of randomness which enables it to discover food accumulations. After a while, one of them will find something digestible and, being social, communicates this to its neighbors. These can then approach the source of food, thus leading to the convergence of the swarm to the source of food. Following this analogy, PSO was largely derived from sociopsychology concept and transferred to optimization [1], where each particle (bird) uses the local information regarding the displacement of its reachable closer neighbors to decide on its own displacement, resulting to complex and adaptive collective behaviors.

Since the inception of PSO technique, a lot of work has been done by researchers to enhance its efficiency in handling optimization problems. One such work is the introduction of linear decreasing inertia weight (LDIW) strategy into the original PSO to control its exploration and exploitation for better performance [2–4]. However, LDIW-PSO algorithm from the literature is known to have the shortcoming of premature convergence in solving complex (multipeak) problems due to lack of enough momentum for particles to do exploitation as the algorithm approaches its terminal point. The challenge of addressing this shortcoming has been on for a long time and has attracted much attention of researchers in the field of global optimization. Consequently upon this, many other inertia weight PSO variants have been proposed [2, 5–16], with different levels of successes. Some of these variants have claimed better performances over LDIW-PSO, thereby making it look weak or inferior. Also, since improving on the performance of PSO is an area which still attracts more researchers, this paper strives to experimentally establish the fact that LDIW-PSO is very much efficient if its parameters, like velocity limits for the particles, are properly set. Using

the experimentally obtained values for particle velocity limits in LDIW-PSO, results show that LDIW-PSO outperformed other PSO variants adopted for comparison.

In the sections that follow, inertia weight PSO technique is described in Section 2, LDIW-PSO and the PSO variants adopted for comparison are reviewed in Section 3, parameter settings were experimentally conducted in Section 4, presentations and discussions of the experimental results of LDIW-PSO and its competing variants are made in Section 5, while Section 6 concludes the paper.

2. Particle Swarm Optimization

This technique is a simple but efficient population-based, adaptive, and stochastic technique for solving simple and complex optimization problems [17, 18]. It does not need the gradient of the problems to work with, so the technique can be employed for a host of optimization problems. In PSO, a swarm of particles (set of solutions) is randomly positioned (distributed) in the search space. For every particle, the objective function determines the food at its place (value of the objective function). Every particle knows its own actual value of the objective function, its own best value (locally best solution), the best value of the whole swarm (globally best solution), and its own velocity.

PSO maintains a single static population whose members are tweaked (adjust slightly) in response to new discoveries about the space. The method is essentially a form of directed mutation. It operates almost exclusively in multidimensional metric, and usually real-valued, spaces. Because of its origin, PSO practitioners tend to refer to candidate solutions not as a population of individuals but as a swarm of particles. Generally, these particles never die [19], but are moved about in the search space by the directed mutation.

Implementing PSO involves a small number of different parameters that regulates the behavior and efficacy of the algorithm in optimizing a given problem. These parameters are particle swarm size, problem dimensionality, particle velocity, inertia weight, particle velocity limits, cognitive learning rate, social learning rate, and the random factors. The versatility of the usage of PSO comes at a price because for it to work well on any problem at hand, these parameters need tuning and this could be very laborious. The inertia weight parameter (popularly represented as ω) has attracted a lot of attentions and seems to be the most important compared with other parameters. The motivation behind its introduction was the desire to better control (or balance) the scope of the (local and global) search and reduce the importance of (or eliminate) velocity clamping, $V_{\max}$, during the optimization process [20–22]. According to [22], the inertia weight was successful in addressing the former objective, but could not completely eliminate the need for velocity clamping. The feature of the divergence or convergence of particles can be controlled only by parameter ω, however, in conjunction with the selection of values for the acceleration constants [22, 23] as well as other parameters.

Each individual in the particle swarm is composed of three n-dimension vectors (current position, previous position, and velocity), where n is the dimensionality of the search space. Thus, in a physical n-dimensional search space, the position and velocity of each particle i are represented as the vectors $X_i = (x_{i1}, \ldots, x_{in})$ and $V_i = (v_{i1}, \ldots, v_{in})$, respectively. In course of movement in the search space looking for the optimum solution of the problem being optimized, the particle's *velocity* and *position* are updated as follows:

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 \left(P\text{best}_i^k - X_i^k\right) + c_2 r_2 \left(G\text{best}_i^k - X_i^k\right), \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \quad (2)$$

where, c_1 and c_2 are acceleration (weighting) factors known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of Pbest (previous best) and Gbest (global previous best); r_1 and r_2 are random numbers between 0 and 1; k is iteration index; ω is inertia weight. It is common that the positions and velocities of particles in the swarm, when they are being updated, are controlled to be within some specified bounds as shown in Algorithms 1 and 2, respectively. An inertia weight PSO algorithm is shown in Algorithm 3.

```
If x_i < min X
  x_i = min X
else if x_i > max X
  x_i = max X
end if
```

Algorithm 1: Particle position clamping.

```
If v_i < min V
  v_i = min V
else if x_i > max V
  v_i = max V
end if
```

Algorithm 2: Particle velocity clamping.

3. A Review of LDIW-PSO and Some of Its Competing PSO Variants

Despite the fact that LDIW-PSO algorithm, from the literature, is known to have a shortcoming of premature convergence in solving complex (multipeak) problems, it may not always be true that LDIW-PSO is as weak or inferior as it has been pictured to be by some PSO variants in the literature [2, 7, 13]. Reviewed below are some of these variants and other variants, though not directly compared with LDIW-PSO in the literature, but have been adopted for comparison with LDIW-PSO.

3.1. Linear Decreasing Inertia Weight PSO (LDIW-PSO). The inertia weight parameter was introduced into the original

Begin Algorithm
Input: function to optimize, f
swarm size, ps
problem dimension, d
search space range, $[\min X, \max X]$
velocity range, $[\min V, \max V]$
Output: x^*: the best value found
Initialize: for all particles in problem space
$x_i = (x_{i1}, \ldots, x_{id})$ and
$v_i = (v_{i1}, \ldots, v_{id})$,
Evaluate $f(x_i)$ in d variables and get $p\text{best}_i$, $(i = 1, \ldots, ps)$
$g\text{best} \leftarrow$ best of $p\text{best}_i$
Repeat
Calculate ω
Update v_i for all particles using (1)
Update x_i for all particles using (2)
Evaluate $f(x_i)$ in d variables and get $p\text{best}_i$, $(i = 1, \ldots, ps)$
If $f(x_i)$ is better than $p\text{best}_i$ then $p\text{best}_i \leftarrow x_i$
If the best of $p\text{best}_i$ is better than $g\text{best}$ then $g\text{best} \leftarrow$ best of $p\text{best}_i$
Until Stopping criteria (e.g., maximum iteration or error tolerance is met)
$x^* \leftarrow g\text{best}$
Return x^*
End Algorithm

Algorithm 3: Inertia weight PSO algorithm.

version of PSO by [20]. By introducing a linearly decreasing inertia weight into the original version of PSO, the performance of PSO has been greatly improved through experimental study [24]. In order to further illustrate the effect of this linearly decreasing inertia weight, [4] empirically studied the performance of PSO. With the conviction that a large inertia weight facilitates a global search while a small inertia weight facilitates a local search, a linearly decreasing inertia weight was used with an initial value of 0.9 and a final value of 0.4. By reason of these values, the inertia weight can be interpreted as the fluidity of the medium in which a particle moves [21], showing that setting it to a relatively high initial value (e.g., 0.9) makes particles move in a low viscosity medium and performs extensive exploration. Gradually reducing it to a much lower value (e.g., 0.4) makes the particle moves in a high viscosity medium and performs more exploitation. The experimental results in [4] showed that the PSO converged quickly towards the optimal positions but slowed down its convergence speed when it is near the optima. Thus, by using the linearly decreasing inertia weight, the PSO lacks global search ability at the end of run even when the global search ability is required to jump out of the local minimum in some cases. As a result, employing adapting strategy for adjusting the inertia weight was suggested to improve PSO's performance near the optima. Towards achieving this, there are many improvements on LDIW-PSO in the literature [2, 3, 16, 24–26], which have made PSO to perform with varying degree of successes. Represented in (3) is the LDIW:

$$\omega_t = (\omega_{\text{start}} - \omega_{\text{end}}) \left(\frac{T_{\max} - t}{T_{\max}} \right) + \omega_{\text{end}}, \tag{3}$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, t is the current iteration number, $T_{\max}$ is the maximum iteration number, and $\omega_t \in [0, 1]$ is the inertia weight value in the tth iteration.

3.2. Chaotic Descending Inertia Weight PSO (CDIW-PSO). Chaos is a nonlinear dynamic system which is sensitive to the initial value. It has the characteristic of ergodicity and stochastic property. Using the idea of chaotic mapping, CDIW-PSO was proposed by [2] as shown in (5) based on logistic mapping in (4). The goal was to improve on the LDIW-PSO to avoid getting into local optimum in searching process by utilizing the merits of chaotic optimization

$$z_{k+1} = \mu \times z_k \times (1 - z_k), \tag{4}$$

where $\mu = 4$ and z_k is the kth chaotic number. The map generates values between 0 and 1, provided that the initial value $z_0 \in (0, 1)$ and that $z_0 \notin (0.0, 0.25, 0.5, 0.75, 1.0)$:

$$\omega_t = (\omega_{\text{start}} - \omega_{\text{end}}) \left(\frac{T_{\max} - t}{T_{\max}} \right) + \omega_{\text{end}} \times z_{k+1}, \tag{5}$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, and rand() is a uniform random number in $[0, 1]$. The experimental results in [2] show that CDIW-PSO outperformed LDIW-PSO in all the test problems used in the experiment in terms of convergence precision, quick convergence velocity, and better global search ability.

3.3. Random Inertia Weight and Evolutionary Strategy PSO (REPSO). This variant proposed in [7] used the idea of simulated annealing and the fitness of particles to design another

inertia weight represented by (6). A cooling temperature was introduced to adjust the inertia weight based on certain probability to facilitate jumping off local optimal solutions.

It was experimentally proven that REPSO is significantly superior LDIW-PSO in terms of convergent speed and accuracy:

$$\omega_t = \begin{cases} \alpha_1 + \dfrac{r}{2.0}, & p \geq r, \\ \alpha_2 + \dfrac{r}{2.0}, & p < r, \end{cases} \tag{6}$$

where $\alpha_1, \alpha_1 \in [0, 1]$ are constants with $\alpha_1 > \alpha_2$ and $r \in U[0, 1]$. The simulated annealing probability is defined as follows:

$$p = \begin{cases} 1, & \min\limits_{1\leq i\leq m} f_i^{t-k} \leq \min\limits_{1\leq i\leq m} f_i^t, \\ \exp\left(-\dfrac{\min_{1\leq i\leq m} f_i^{t-k} - \min_{1\leq i\leq m} f_i^t}{T_t}\right), & \min\limits_{1\leq i\leq m} f_i^{t-k} > \min\limits_{1\leq i\leq m} f_i^t, \end{cases} \tag{7}$$

where m is the number of particles, f_i^t is the fitness value of particle i in the tth iteration, and the adaptive cooling temperature in the tth iteration, T_t, is defined as shown in (8):

$$T_t = \frac{f_{\text{avg}}^t}{f_{\text{best}}^t} - 1, \tag{8}$$

where f_{best}^t is the current best fitness value, and f_{avg}^t which is defined in (9), is the average fitness value in the tth iteration:

$$f_{\text{avg}}^t = \frac{1}{m}\sum_{i=1}^{m} f_i^t. \tag{9}$$

The combined efforts of the simulated annealing idea and fitness variance of particles improved the global search ability of PSO. In all the experiments performed, REPSO was recorded superior to LDIW-PSO in convergence velocity and precision.

3.4. Dynamic Adaptive Particle Swarm Optimization (DAPSO). DAPSO was introduced by [3] with the aim of proffering solution to the PSO premature convergence problem associated with typical multipeak, high dimensional function optimization problems so as to improve its global optimum and convergence speed. A dynamic adaptive strategy was introduced into the variant to adjust the inertia weight value based on the current swarm diversity and congregate degree as well as the impact on the search performance of the swarm. The experimental results recorded showed that DAPSO was more effective compared with LDIW-PSO. The inertia weight formula that was used is represented in (10):

$$\omega_t = \omega_{\min} + (\omega_{\max} - \omega_{\min}) \times F_t \times \varphi_t, \tag{10}$$

where $\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum inertia weight values, t is the current number of iterations, the diversity function F_t and adjustment function φ_t, both in the tth iteration, are represented in (11) and (12), respectively:

$$F_t = 1 - \frac{2}{\pi}\arctan(E), \tag{11}$$

where E is the group fitness as shown in (13):

$$\varphi_t = e^{(-t^2/(2\sigma^2))}, \tag{12}$$

where $\sigma = T/3$ and T are the total numbers of iterations:

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - f_{\text{avg}}\right)^2, \tag{13}$$

where N is the swarm size, $f(x_i)$ is the fitness of particle i, and f_{avg} represented in (14) is the current average fitness of the swarm:

$$f_{\text{avg}} = \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{14}$$

3.5. Adaptive Particle Swarm Optimization (APSO). This PSO variant was proposed in [5], in which an adaptive mutation mechanism and a dynamic inertia weight were incorporated into the conventional PSO method. These mechanisms were employed to enhance global search ability and convergence speed and to increase accuracy, while the mutation mechanism affected the particle position updating formula, the dynamic inertia weight affected the inertia weight formula shown in (15). Though APSO was not compared with LDIW-PSO, it outperformed all its competitors as evidenced by all the experimental results:

$$\omega_t = 0.5\left\{1 + \tanh\left[\frac{1}{\alpha} \times F\left(P_{gd}^t\right)\right]\right\}, \tag{15}$$

where $F(P_{gd}^t)$ is the fitness of current best solution in the tth iteration, and the parameter α is predefined which could be set equal to the fitness of the best particle in the initial population. For the updating of the particle's position, when a particle is chosen for mutation, a Gaussian random disturbance was added as depicted in (16):

$$x_{ij} = x_{ij} + M \times \beta_{ij}, \tag{16}$$

where x_{ij} is the ith component of the jth particle, β^{ij} is a random variable with Gaussian distribution with zero mean and unit variance, and M is a variable step size which dynamically decreases according to current best solution fitness. M is defined in tth iteration according to

$$M_t = x_{\max} \times \tanh\left[\frac{1}{\alpha} \times F\left(P_{gd}^t\right)\right]. \tag{17}$$

3.6. Dynamic Nonlinear and Dynamic Logistic Chaotic Map PSO (DLPSO2). In [11], two types of variants were proposed to solve the premature convergence problem of PSO. In this variant, two dynamic nonlinear methods and logistic chaotic map were used to adjust the inertia weight in parallel

Table 1: Settings for parameter δ in LDIW-PSO.

Problem	f_1	f_2	f_3	f_4	f_5	f_6
δ	0.05	0.0075	0.05	0.015	0.075	0.015

according to different fitness values. One of the dynamic nonlinear inertia weights is represented by (18) and (19), while the other is represented by (20) and (21). They are meant to achieve tradeoff between exploration and exploitation:

$$dn = dn_{\min} + (dn_{\max} - dn_{\min})\left(\frac{\text{iter}}{\text{iter}_{\max}}\right), \quad (18)$$

$$\omega = \omega_{\min} + (\omega_{\max} - \omega_{\min})\left(\frac{\text{iter}}{\text{iter}_{\max}}\right)^{dn}, \quad (19)$$

$$dn = dn_{\max} - (dn_{\max} - dn_{\min})\left(\frac{\text{iter}}{\text{iter}_{\max}}\right), \quad (20)$$

$$\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min})\left(\frac{\text{iter}}{\text{iter}_{\max}}\right)^{dn}, \quad (21)$$

where dn is the dynamic nonlinear factor, ω is the inertia weight, $\omega_{\max}$ and $\omega_{\min}$ are the maximum and minimum values of ω, respectively, $dn_{\max}$ and $dn_{\min}$ are the maximum and minimum values of dn, respectively, and iter and $\text{iter}_{\max}$ are the current iteration numbers and the maximum iteration number, respectively.

A dynamic logistic chaotic map in (4) was incorporated into the PSO variant inertia weight as shown in (23) to enrich searching behaviors and avoid being trapped into local optima:

$$\alpha = \alpha_{\max} - (\alpha_{\max} - \alpha_{\min})\left(\frac{\text{iter}}{\text{iter}_{\max}}\right), \quad (22)$$

$$\omega = \alpha + (1 - \alpha)\,\text{Lmap}, \quad (23)$$

where α is the dynamic chaotic inertia weight adjustment factor, $\alpha_{\max}$ and $\alpha_{\min}$ are the maximum and minimum values of α, respectively, and Lmap is the result of logistic chaotic map. In this variant, using (19) and (23) was labeled DLPSO1, while using (21) and (23) was captioned DLPSO2.

For the purpose of achieving a balance between global exploration and local exploitation and also avoiding premature convergence, (19), (21), and (23) were mixed together to dynamically adjust the inertia weight of the variant as shown in Algorithm 4, where f_i is the fitness value of particle i and f_{avg} is the average fitness value of the swarm. Experiments and comparisons showed that the DLPSO2 outperformed several other well-known improved particle swarm optimization algorithms on many famous benchmark problems in all cases.

3.7. Discussions. LDIW-PSO is relatively simple to implement and fast in convergence. When [4] experimentally ascertained that LDIW-PSO is prone to premature convergence, especially when solving complex multimodal optimization problems, a new area of research was opened up for improvements on inertia weight strategies in PSO, and LDIW-PSO became a popular yard stick for many other variants.

```
if f_i ≤ f_avg
   compute ω using (19) or (21)
   elseif f_i > f_avg
   compute ω using (23)
end if
```

Algorithm 4

From the variants described previously, there are ample expectations that they should outperform LDIW-PSO judging by the various additional strategies introduced into the inertia weight strategies used by them. For example, CDIW-PSO introduced chaotic optimization characteristic, REPSO introduced a combined effort of simulated annealing idea and fitness variance of particles, DAPSO introduced a dynamic adaptive strategy based on swarm diversity function, APSO introduced an adaptive mutation to the particle positions and made the inertia weight dynamic based on the best global fitness, while DLPSO2 used different formulas coupled with chaotic mapping. The general aims of remedying the problem of premature convergence by these variants were not achieved, rather they only struggled to move a bit further than LDIW-PSO in trying to optimize the test problems because a total solution to this problem is for an algorithm to escape all possible local optima and obtain the global optimum. With this, it is possible that LDIW-PSO was subjected to settings, for example, the particles velocity limits [24], which were not appropriate for it to operate effectively.

4. Testing with Benchmark Problems

To validate the claim in this paper, 6 different experiments were performed for the purpose of comparing the LDIW-PSO with 6 other different PSO variants, namely, AIW-PSO, CDIW-PSO, REPSO, SA-PSO, DAPSO, and APSO. Different experiments, relative to the competing PSO variants, used different set of test problems which were also used to test LDIW-PSO. Brief descriptions of these test problems are given next. Additional information on these problems can be found in [27–29]. The application software was developed in Microsoft Visual C# programming language.

4.1. Benchmark Problems. Six well studied benchmark problems in the literature were used to test the performance of LDIW-PSO, AIW-PSO, CDIW-PSO, REPSO, SA-PSO, DAPSO, and APSO. These problems were selected because their combinations were used to validate the competing PSO variants in the respective literature referenced. The test problems are Ackley, Griewank, Rastrigin, Rosenbrock, Schaffer's f6, and Sphere.

The Ackley problem is multimodal and nonseparable. It is a widely used test problem, and it is defined in (24). The

Table 2: Test problems search and initialization ranges for the PSO variants.

Label	CDIW-PSO	REPSO	DAPSO	APSO	DLPSO2
f_1	—	—	$[-32, 32]$	—	$[-32, 32]$
f_2	$[-600, 600]$	$[-600, 600]$	$[-600, 600]$	$[-600, 600]$	$[-600, 600]$
f_3	$[-5.12, 5.12]$	$[-10, 10]$	$[-5.12, 5.12]$	$[-5.12, 5.12]$	$[-10, 10]$
f_4	$[-30, 30]$	$[-100, 100]$	—	$[-30, 30]$	—
f_5	$[-100, 100]$	$[-10, 10]$	—	—	$[-1.0, 1.0]$
f_6	$[-100, 100]$	$[-10, 10]$	—	—	$[-100, 100]$

Table 3: Goals for the test problems in CDIW-PSO.

Label	f_2	f_3	f_4	f_5	f_6
Goal	0.05	50.0	100.0	0.00001	0.01

global minimum $f_1(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$, and the number of local minima is not known:

$$f_1(\vec{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e. \quad (24)$$

The Griewank problem is similar to that of Rastrigin. It is continuous, multimodal scalable, and nonseparable with many widespread local minima regularly distributed. The complexity of the problem increases with its dimensionality. Its global minimum $f_2(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$, and the number of local minima for arbitrary n is not known, but in the two-dimensional case, there are some 500 local minima. This problem is represented by

$$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1. \quad (25)$$

The Rastrigin problem represented in (26) is continuous, multimodal, scalable, and separable with many local minima arranged in a lattice-like configuration. It is based on the Sphere problem with the addition of cosine modulation so as to produce frequent local minima. There are about 50 local minima for two-dimensional case, and its global minimum $f_3(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$:

$$f_3(\vec{x}) = \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right). \quad (26)$$

Shown in (27) is the Rosenbrock problem. It is continuous, unimodal, scalable, and nonseparable. It is a classic optimization problem also known as banana function, the second function of De Jong, or extended Rosenbrock function. Its global minimum $f_4(\vec{x}) = 0$ obtainable at $\vec{x} = 1$, lies inside a long narrow, parabolic shaped valley. Though it looks simple to solve, yet due to a saddle point it is very difficult to converge to the global optimum:

$$f_4(\vec{x}) = \sum_{i=1}^{d-1}\left(100\left(x_{i+1} - x_i^2\right)^2\right) + (x_i - 1)^2. \quad (27)$$

The Schaffer's f6 problem represented in (28) is 2-dimensional, continuous, multimodal, and nonseparable with unknown number of many local minima. Its global minimum $f_5(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$:

$$f_5(\vec{x}) = \sum_{i=1}^{d-1}\left(0.5 + \frac{\sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{\left(0.001\left(x_{i+1}^2 + x_i^2\right) + 1\right)^2}\right). \quad (28)$$

The Sphere problem also known as the first De Jong function is continuous, convex, unimodal, scalable, and separable. It is one of the simplest test benchmark problems. Its global minimum $f_6(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$, and the problem is represented by

$$f_6(\vec{x}) = \sum_{i=1}^{d} x_i^2. \quad (29)$$

4.2. Parameter Settings. The limits of particle velocity could negatively affect the performance of PSO algorithm if it is not properly set. As a result, different work has been done to determine the velocity limits of particles in order to improve on the performance of PSO. Researches in this direction are [4, 24, 30] the three major methods that appear in the literature, for computing the velocity clamping ($V_{\min}$ and $V_{\max}$) are (i) multiplying the search space range with certain percentage (δ); that is, $V_{\max} = \delta(X_{\max} - X_{\min})$ and $V_{\min} = -V_{\max}$; (ii) multiplying both the minimum and maximum limits of the search space separately with certain percentage (δ); that is, $V_{\max} = \delta(X_{\max})$ and $V_{\min} = \delta(X_{\min})$; (iii) assigning the search space upper limit to $V_{\max}$. It is obvious from (i) and (ii) that the parameter δ is very important. As a result, different values have been used by different authors [5, 6, 13, 30] for δ to determine velocity clamping for particles.

In trying to substantiate the fact that LDIW-PSO is not as weak or inferior as many authors claimed it to be, an experiment was conducted to investigate the effect of the parameter δ on the performance of LDIW-PSO using the benchmark problems described previously. The results were used as a guide to set δ in LDIW-PSO before embarking on some experimental comparison, between it and some other

Table 4: Experimental results for LDIW-PSO compared with CDIW-PSO.

Criteria	Griewank		Rastrigin		Rosenbrock		Schaffer's f6		Sphere	
	CDIW-PSO	LDIW-PSO	CDIW-PSO	LDIW-PSO	CDIW-PSO	LDIW-PSO	CDIW-PSO	LDIW-PSO	CDIW-PSO	LDIW-PSO
Mean fitness	0.014773	**0.007609**	40.044561	**33.055877**	44.305058	**31.148789**	0.007732	**0.000117**	0.000092	**0.000000**
Std. Dev.	0.002959	**0.008439**	**8.028912**	10.498048	**8.861012**	20.832263	0.001546	**0.001058**	0.000016	**0.000000**
SR (%)	96.2	**100**	83.6	**92.8**	**99.6**	98.0	22.0	**98.6**	100	100

Table 5: Experimental results for LDIW-PSO compared with REPSO.

Iteration	Griewank[1]		Rastrigin		Rosenbrock[2]		Sphere	
	REPSO	LDIW-PSO	REPSO	LDIW-PSO	REPSO	LDIW-PSO	REPSO	LDIW-PSO
50	—	—	—	—	—	—	—	—
100	**0.6705**	0.7859	**30.7320**	44.2732	—	—	0.00671	**0.00493**
200	**0.4922**	0.6437	—	—	—	—	—	—
300	**0.2487**	0.5607	—	—	—	—	**2.1142e − 04**	2.9792e − 04
400	**0.2345**	0.4318	20.6671	**16.5414**	—	—	—	—
500	**0.1658**	0.3185	17.3751	**10.4621**	570.7681	**352.1663**	7.1144e − 05	**9.1853e − 07**
800	—	—	15.5611	**3.9143**	—	—	6.8751e − 06	**5.8431e − 17**
1000	0.1461	**0.0967**	10.8120	**3.2609**	300.1407	**218.9924**	5.6367e − 07	**1.2425e − 28**
1500	0.1353	**0.0842**	—	—	260.8421	**138.2756**	—	—
2000	0.1089	**0.0794**	—	—	170.2157	**79.9941**	—	—
3000	—	—	—	—	60.4418	**21.5586**	—	—

[1]This problem is slightly different from the one in (25).
[2]This problem is slightly different from the one in (27).

PSO variants described previously to prove that LDIW-PSO is superior to many of the variants that have been claimed to be better that it. The results of the experiments are listed in the Appendix. Using the results as guide, the value of δ was set in LDIW-PSO for the various test problems as listed in Table 1. However, δ was set to 0.015 for f_2 in Experiment 2 and 0.25 for f_3 in Experiments 2 and 5.

4.3. Experimental Setup. The settings for the different experiments carried out for the comparisons are described next one after the other. In all the experiments, LDIW-PSO was subjected to the settings of its competitors as recorded in the literature. For LDIW-PSO, $c_1 = c_2 = 2.0$, $\omega_{\max} = 0.9$, $\omega_{\min} = 0.4$, $V_{\min} = \delta X_{\min}$, and $V_{\max} = \delta X_{\max}$. Table 2 shows the respective search and initialization ranges for all the algorithms, the symbol "–" means that the corresponding test problem was not used by the algorithm under which the symbol appears.

Experiment 1. The purpose of this experiment was to compare LDIW-PSO with CDIW-PSO [2]. All the test problems described previously were used in this experiment, except f_1. The dimension for f_5 was 2, while it was 30 for others. The maximum numbers of iterations were set to 1500 with swarm size of 20, and the experiment was repeated 500 times. Stated in Table 3 are the set goals (criteria) of success for each of the problems.

Experiment 2. The purpose of this experiment was to compare LDIW-PSO with REPSO [7]. All the test problems in Table 1 except f_1 were used. The dimension for f_5 was 2, while it was 10 for others. The performances of the algorithms were considered at different number of iterations as shown in Section 5, in line with what is recorded in the literature [7]. The swarm size used was 30, and the experiment was repeated 50 times.

Experiment 3. The purpose of this experiment was to compare LDIW-PSO with DAPSO [13]. Test problems $f_1 - f_3$ were used with four different problem dimensions of 20, 30, 40, and 50. The maximum number of iterations and swarm size was set to 3000 and 30, respectively, and the experiment was repeated 50 times.

Experiment 4. The purpose of this experiment was to compare LDIW-PSO with APSO [5]. f_2, f_3, and f_4 were used as test problems with three different problem dimensions of 10, 20, and 30. The respective maximum numbers of iterations associated with these dimensions are 1000, 1500, and 2000, respectively. The experiment was carried out over three different swarm sizes, 20, 40, and 80 for each problem dimension, and the experiment was repeated 30 times.

Experiment 5. This experiment compared LDIW-PSO with DLPSO2 [11]. All the test problems except f_4 were used in the experiment with two different problem dimensions of 2 (for

Table 6: Experimental results for LDIW-PSO compared with DAPSO.

Dim	Ackley		Griewank		Rastrigin	
	DAPSO	LDIW-PSO	DAPSO	LDIW-PSO	DAPSO	LDIW-PSO
20	3.906209$e-014$	**8.970602e − 015**	8.605280$e-002$	**1.649481e − 002**	2.159059$e+001$	**2.040020e + 001**
30	4.159541$e-008$	**1.527799e − 010**	2.583338$e-002$	**9.258783e − 003**	3.263463$e+001$	**2.996404e + 001**
40	7.046093$e-005$	**2.578715e − 007**	1.087868$e-002$	**4.875733e − 003**	**3.890287e + 001**	4.109865$e+001$
50	1.025568$e-003$	**1.629095e − 005**	1.346732$e-002$	**4.335978e − 003**	4.823559$e+001$	**4.606947e + 001**

Table 7: Experimental results for LDIW-PSO compared with APSO.

Swarm size	Dim	Maximum iteration	Griewank		Rastrigin		Rosenbrock	
			APSO	LDIW-PSO	APSO	LDIW-PSO	APSO	LDIW-PSO
20	10	1000	**0.0983**	0.2347	**5.1565**	12.4602	5.8467	**4.3695**
	20	1500	0.0237	**0.0150**	**16.0456**	27.6708	47.9842	**19.1223**
	30	2000	0.0117	**0.0103**	42.2325	**33.2050**	100.4528	**29.3482**
40	10	1000	**0.0952**	0.2231	**2.9468**	10.5713	4.5431	**3.9145**
	20	1500	**0.0201**	0.0211	**15.3678**	19.3199	38.3464	**16.5186**
	30	2000	0.0105	**0.0099**	33.7538	**26.3453**	72.5473	**26.9638**
80	10	1000	**0.0689**	0.1294	**2.0457**	9.0800	**4.1680**	6.5127
	20	1500	0.0199	**0.0184**	**10.0563**	16.4368	27.9547	**17.6043**
	30	2000	0.0102	**0.0080**	25.3473	**23.2303**	69.0609	**24.6653**

Table 8: Experimental results for LDIW-PSO compared with DLPSO2.

Criteria	Best fitness	Mean fitness	Std. Dev.
Ackley			
DLPSO2	8.6209$e-06$	0.4743	0.6527
LDIW-PSO	**2.0441e − 07**	**0.0000**	**0.0000**
Griewank			
DLPSO2	7.7589$e-06$	0.0086	0.0114
LDIW-PSO	**3.5694e − 13**	**0.0083**	**0.0088**
Rastrigin			
DLPSO2	−2	−2	0
LDIW-PSO	−2	−2	0
Schaffer's f6			
DLPSO2	7.5206$e-07$	5.6300$e-06$	2.8969$e-06$
LDIW-PSO	**0.0000e + 00**	**0.0000e + 00**	**0.0000e + 00**
Sphere			
DLPSO2	7.6941$e-06$	9.5001$e-06$	4.9557$e-07$
LDIW-PSO	**4.1289e − 14**	**0.0000e + 00**	**0.0000e + 00**

f_3 and f_5) and 30 (for f_1, f_2, and f_6). The maximum number of iterations was set to 2000 and swarm sizes to 20, and the experiment was repeated 20 times.

5. Results and Discussions

Presented in Tables 4–8 are the results obtained for all the experiments. The results for all the competing PSO variants were obtained from the respective referenced papers, and they are presented here the way they were recorded. Thus, the recording of the results for LDIW-PSO was patterned after them. In each of the tables, bold values represent the best results. In the tables, mean best fitness (solution) is a measure of the precision that the algorithm can get within a given number of iterations, standard deviation (Std. Dev.) is a measure of the algorithm's stability and robustness, while success rate (SR) [31] is the rate at which an algorithm obtains optimum fitness result in the criterion range during a given number of independent runs.

Experiment 1 (comparison of LDIW-PSO with CDIW-PSO). The results in Table 4 clearly reveal a great difference in performance between LDIW-PSO and CDIW-PSO [2]. The results are compared based on the final accuracy of the averaged best solutions, success rate (SR), and standard deviation (Std. Dev.) of the best solutions. In all the test problems, the result indicates that LDIW-PSO can get better optimum fitness result, showing better convergence precision. LDIW-PSO is also more stable and robust compared with CDIW-PSO, because its standard deviation is comparatively lesser in three of the test problems. Besides, LDIW-PSO has better global search ability and could easily get out of local optima than CDIW-PSO.

Experiment 2 (comparison of LDIW-PSO with REPSO). In Table 5, the comparison between LDIW-PSO and REPSO was based on the final accuracy of the averaged best solutions relative to the specified number of iterations and convergence speed as recorded in [7]. From the results, REPSO appears to converge faster in Griewank and Rastrigin at the beginning but was overtaken by LDIW-PSO which eventually converged faster and had better accuracy. In Rosenbrock and Sphere problems, LDIW-PSO had better convergence speed and accuracy in comparison with REPSO. The symbol "—" means that the corresponding iteration number was not considered for the test problem under which the symbol appears.

Table 9: Different values of parameter δ and respective mean best fitness for Griewank test problem.

δ	Dimension 10		Dimension 30		Dimension 50	
	Size = 20	Size = 30	Size = 20	Size = 30	Size = 20	Size = 30
1.0	9.913$e-02$	9.125$e-02$	1.157$e+01$	5.607$e+00$	6.269$e+01$	3.941$e+01$
0.75	9.645$e-02$	8.825$e-02$	3.088$e+00$	1.451$e-02$	1.519$e+01$	6.875$e+00$
0.5	9.983$e-02$	9.018$e-02$	1.972$e-01$	1.601$e-02$	2.003$e+00$	5.522$e-01$
0.25	1.002$e-01$	**2.925e−02**	1.602$e-02$	1.458$e-02$	1.200$e-02$	9.885$e-03$
0.15	9.772$e-02$	9.276$e-02$	1.556$e-02$	1.450$e-02$	9.925$e-03$	8.654$e-03$
0.1	1.044$e-01$	9.141$e-02$	1.489$e-02$	1.564$e-02$	1.027$e-02$	9.339$e-03$
0.075	1.064$e-01$	1.006$e-01$	1.328$e-02$	1.389$e-02$	8.937$e-03$	7.963$e-03$
0.05	1.011$e-01$	9.417$e-02$	1.521$e-02$	1.580$e-02$	8.224$e-03$	7.821$e-03$
0.025	9.682$e-02$	8.738$e-02$	1.604$e-02$	1.668$e-02$	7.108$e-03$	7.354$e-03$
0.015	**9.028e−02**	8.648$e-02$	1.379$e-02$	1.444$e-02$	5.719$e-03$	6.226$e-03$
0.01	1.274$e-01$	1.265$e-01$	1.148$e-02$	1.141$e-02$	5.005$e-03$	4.768$e-03$
0.0075	2.251$e-01$	2.078$e-01$	**7.160e−03**	**7.595e−03**	4.237$e-03$	**4.021e−03**
0.005	5.546$e-01$	3.751$e-01$	8.006$e-03$	8.030$e-03$	**4.025e−03**	4.526$e-03$
0.0025	1.258$e+00$	6.833$e-01$	1.203$e-02$	1.218$e-02$	6.808$e-03$	6.013$e-03$
0.0015	1.895$e+01$	9.642$e-01$	1.415$e-02$	1.434$e-02$	7.226$e-03$	7.419$e-03$
0.001	4.061$e+00$	2.083$e+00$	1.366$e-02$	1.622$e-02$	7.184$e-03$	7.462$e-03$

Table 10: Different values of parameter δ and respective mean best fitness for Rastrigin test problem.

δ	Dimension 10		Dimension 30		Dimension 50	
	Size = 20	Size = 30	Size = 20	Size = 30	Size = 20	Size = 30
1.0	4.551$e+00$	**3.400e+00**	9.959$e+01$	8.462$e+01$	2.694$e+02$	2.361$e+02$
0.75	**4.537e+00**	3.619$e+00$	6.924$e+01$	5.866$e+01$	1.935$e+02$	1.729$e+02$
0.5	4.646$e+00$	3.476$e+00$	5.253$e+01$	4.282$e+01$	1.330$e+02$	1.151$e+02$
0.25	6.484$e+00$	5.247$e+00$	4.534$e+01$	4.197$e+01$	8.943$e+01$	8.462$e+01$
0.15	1.043$e+01$	9.013$e+00$	4.142$e+01$	3.798$e+01$	7.204$e+01$	6.590$e+01$
0.1	1.149$e+01$	9.470$e+00$	3.702$e+01$	3.380$e+01$	6.183$e+01$	5.653$e+01$
0.075	1.077$e+01$	9.397$e+00$	3.328$e+01$	2.917$e+01$	5.394$e+01$	4.824$e+01$
0.05	1.162$e+01$	1.022$e+01$	**3.302e+01**	**2.943e+01**	**5.370e+01**	**4.704e+01**
0.025	1.373$e+01$	1.160$e+01$	3.607$e+01$	3.194$e+01$	5.474$e+01$	4.860$e+01$
0.015	1.387$e+01$	1.159$e+01$	3.893$e+01$	3.521$e+01$	5.762$e+01$	5.087$e+01$
0.01	1.431$e+01$	1.221$e+01$	4.010$e+01$	3.565$e+01$	5.995$e+01$	5.390$e+01$
0.0075	1.475$e+01$	1.213$e+01$	4.164$e+01$	3.692$e+01$	6.256$e+01$	5.476$e+01$
0.005	1.868$e+01$	1.398$e+01$	4.300$e+01$	3.663$e+01$	6.451$e+01$	5.464$e+01$
0.0025	3.337$e+01$	2.507$e+01$	7.294$e+01$	4.917$e+01$	9.215$e+01$	6.073$e+01$
0.0015	4.794$e+01$	4.027$e+01$	1.168$e+02$	7.803$e+01$	1.396$e+02$	8.922$e+01$
0.001	5.792$e+01$	5.220$e+01$	1.898$e+02$	1.548$e+02$	2.102$e+02$	1.390$e+02$

Experiment 3 (comparison of LDIW-PSO with DAPSO). The results for DAPSO were obtained from [13]. Comparing these results with that of LDIW-PSO were measured using the final accuracy of the respective mean best solutions across the different problems dimensions as shown in Table 6. In all the problems and dimensions except dimension 40 of Rastrigin, LDIW-PSO outperformed DAPSO getting better fitness quality and precision. This is a clear indication that LDIW-PSO has better global search ability and is not easily trapped in local optima compared with DAPSO.

Experiment 4 (comparison of LDIW-PSO with APSO). Recorded in Table 7 are the results for LDIW-PSO and APSO [5] over different swarm sizes, dimensions, and iterations. The basis for comparison is the final accuracy and quality of their mean best fitness. The two variants put up a good competition. In Griewank and Rastrigin, APSO performed better in smaller dimensions, while LDIW-PSO performed better in higher dimensions. But in Rosenbrock, LDIW-PSO outperformed APSO in locating better solutions to the problem.

Experiment 5 (comparison of LDIW-PSO with DLPSO2). The results for LIDIW-PSO and DLPSO2 [11] in Table 8 are compared based on the best fitness, mean best fitness, convergence speed, as well as standard deviation (Std. Dev.) of

Table 11: Different values of parameter δ and respective mean best fitness for Rosenbrock test problem.

δ	Dimension 10		Dimension 30		Dimension 50	
	Size = 20	Size = 30	Size = 20	Size = 30	Size = 20	Size = 30
1.0	1.165*e* + 04	1.040*e* + 04	1.851*e* + 05	2.873*e* + 04	3.075*e* + 06	1.148*e* + 06
0.75	6.020*e* + 03	4.020*e* + 03	2.009*e* + 04	1.711*e* + 04	8.240*e* + 05	1.837*e* + 05
0.5	2.585*e* + 03	2.189*e* + 03	1.128*e* + 04	8.214*e* + 03	1.175*e* + 04	1.360*e* + 04
0.25	1.872*e* + 01	5.571*e* + 00	4.307*e* + 02	4.445*e* + 02	2.315*e* + 03	1.056*e* + 03
0.15	1.075*e* + 01	4.229*e* + 00	4.910*e* + 01	4.750*e* + 01	1.156*e* + 02	9.710*e* + 01
0.1	4.798*e* + 00	4.241*e* + 00	4.248*e* + 01	4.147*e* + 01	9.217*e* + 01	8.699*e* + 01
0.075	**4.680e + 00**	**4.099e + 00**	4.531*e* + 01	3.607*e* + 01	1.073*e* + 02	7.701*e* + 01
0.05	5.182*e* + 00	4.534*e* + 00	3.453*e* + 01	3.282*e* + 01	6.858*e* + 01	6.383*e* + 01
0.025	5.770*e* + 00	5.598*e* + 00	3.148*e* + 01	3.035*e* + 01	5.450*e* + 01	5.215*e* + 01
0.015	7.818*e* + 00	6.800*e* + 00	**2.956e + 01**	**2.832e + 01**	**5.207e + 01**	5.218*e* + 01
0.01	7.748*e* + 00	6.480*e* + 00	2.962*e* + 01	2.891*e* + 01	5.487*e* + 01	**5.154e + 01**
0.0075	8.085*e* + 00	7.945*e* + 00	2.998*e* + 01	2.948*e* + 01	5.505*e* + 01	5.164*e* + 01
0.005	6.491*e* + 00	6.896*e* + 00	3.134*e* + 01	3.015*e* + 01	5.544*e* + 01	5.263*e* + 01
0.0025	7.943*e* + 01	7.682*e* + 00	3.052*e* + 01	2.915*e* + 01	5.656*e* + 01	5.163*e* + 01
0.0015	5.003*e* + 01	1.408*e* + 01	3.095*e* + 01	2.672*e* + 01	5.398*e* + 01	5.174*e* + 01
0.001	2.417*e* + 04	3.426*e* + 03	3.020*e* + 01	2.949*e* + 01	5.614*e* + 01	5.222*e* + 01

Table 12: Different values of parameter δ and respective mean best fitness for Sphere test problem.

δ	Dimension 10		Dimension 30		Dimension 50	
	Size = 20	Size = 30	Size = 20	Size = 30	Size = 20	Size = 30
1.0	1.043*e* − 20	3.679*e* − 23	1.140*e* + 03	5.400*e* + 02	7.380*e* + 03	4.400*e* + 03
0.75	9.490*e* − 21	1.554*e* − 23	1.600*e* + 02	4.000*e* + 01	1.460*e* + 03	7.600*e* + 02
0.5	5.108*e* − 21	1.048*e* − 23	1.349*e* − 08	4.015*e* − 10	1.000*e* + 02	2.000*e* + 01
0.25	8.561*e* − 22	5.859*e* − 24	3.547*e* − 09	6.110*e* − 11	1.538*e* − 05	4.976*e* − 07
0.15	5.304*e* − 21	9.144*e* − 25	1.503*e* − 09	2.963*e* − 11	6.952*e* − 06	2.114*e* − 07
0.1	6.679*e* − 23	1.203*e* − 24	4.432*e* − 10	1.193*e* − 11	2.224*e* − 06	7.656*e* − 08
0.075	8.577*e* − 23	2.149*e* − 25	2.397*e* − 10	8.813*e* − 12	1.306*e* − 06	4.954*e* − 08
0.05	3.921*e* − 23	1.794*e* − 25	1.147*e* − 10	3.490*e* − 12	5.098*e* − 07	2.235*e* − 08
0.025	1.006*e* − 23	4.835*e* − 26	2.596*e* − 11	7.592*e* − 13	1.620*e* − 07	6.654*e* − 09
0.015	2.466*e* − 24	1.795*e* − 26	1.349*e* − 11	2.364*e* − 13	5.689*e* − 08	2.222*e* − 09
0.01	1.022*e* − 24	4.326*e* − 27	3.998*e* − 12	1.245*e* − 13	3.983*e* − 08	8.837*e* − 10
0.0075	9.942*e* − 25	3.991*e* − 27	2.758*e* − 12	7.017*e* − 14	1.115*e* − 08	5.786*e* − 10
0.005	**6.363e − 25**	**2.300e − 27**	1.449*e* − 12	3.061*e* − 14	1.116*e* − 08	2.034*e* − 10
0.0025	2.003*e* − 23	1.376*e* − 26	**3.638e − 13**	**9.420e − 15**	1.592*e* − 09	6.778*e* − 11
0.0015	4.469*e* − 08	2.962*e* − 08	7.378*e* − 13	1.254*e* − 14	**1.062e − 09**	3.130*e* − 11
0.001	2.900*e* + 02	9.887*e* + 01	5.711*e* − 02	8.265*e* − 13	2.563*e* − 09	**2.755e − 11**

the best solutions. In Rastrigin, the two algorithms have equal performances. However, in other test problems, the result indicates that LDIW-PSO can get better optimum fitness result, showing better convergence speed. LDIW-PSO is also more stable and robust compared with DLPSO2, because its standard deviation is comparatively smaller in all the test problems. Besides, LDIW-PSO demonstrated better global search ability and getting out of local optima than DLPSO2.

6. Conclusion

Motivated by the superiority claims by some PSO variants over LDIW-PSO in terms of performance, a number of experiments were performed in this paper to empirically verify some of these claims. Firstly, an appropriate (approximate) percentage of the test problems search space limits were obtained to determine the particle velocity limits for LDIW-PSO. Secondly, these values were used in the implementation of LDIW-PSO for some benchmark optimization problems and the results obtained compared with that of some PSO variants that have previously claimed superiority in performance. LDIW-PSO performed better than these variant. The performances of the two other recent PSO variants with different inertia weight strategies were also compared with LDIW-PSO on similar problems with the latter showing competitive advantage. This work has therefore

Table 13: Different values of parameter δ and respective mean best fitness for Schaffer's f6 test problem.

δ	Dimension 2	
	Size = 20	Size = 30
1.0	1.342$e-03$	5.446$e-04$
0.75	2.392$e-03$	9.335$e-04$
0.5	2.052$e-03$	7.651$e-04$
0.25	1.387$e-03$	7.212$e-04$
0.15	7.756$e-04$	2.731$e-04$
0.1	6.816$e-04$	1.847$e-04$
0.075	**4.865e − 04**	1.749$e-04$
0.05	6.413$e-04$	**1.612e − 04**
0.025	4.275$e-03$	2.740$e-03$
0.015	5.625$e-03$	3.129$e-03$
0.01	4.726$e-03$	2.993$e-03$
0.0075	4.594$e-03$	2.683$e-03$
0.005	5.663$e-03$	3.327$e-03$
0.0025	5.940$e-03$	4.760$e-03$
0.0015	7.582$e-03$	5.449$e-03$
0.001	7.776$e-03$	6.092$e-03$

showed that with good experimental setting, LDIW-PSO will perform competitively with similar variants. Precious claims of inferior performance might therefore be due to some unfavourable experimental settings. The Appendix provides further simulation results that can provide useful hints for deciding the setting velocity threshold for particles for LDIW-PSO.

Appendix

Tables 9, 10, 11, 12, and 13 show the results of LDIW-PSO in optimizing some benchmark problems so as to determine a suitable value for δ that was used to set the velocity limits for the particles. The experiments were repeated 500 times for each of the problems. Two different swarm sizes of 20 and 30 were used for each of the three different problem dimensions 10, 30, and 50. The respective number of iterations that was used with the dimensions is 1000, 1500, and 2000. The LDIW strategy was decreased from 0.9 to 0.4 in course of searching for solution to the problem [7, 10–12, 27], the acceleration constants (c_1 and c_2) were set to 2.0, and $V_{\max} = \delta(X_{\max})$ and $V_{\min} = \delta(X_{\min})$. In the tables, bold values represent the best mean fitness value.

Acknowledgment

Thanks are due to the College of Agricultural Science, Engineering and Sciences, University of Kwazulu-Natal, South Africa, for their support towards this work through financial grant.

References

[1] R. C. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[2] Y. Feng, G. F. Teng, A. X. Wang, and Y. M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, p. 475, Kumamoto, Japan, September 2007.

[3] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, Sanya, China, April 2009.

[4] Y. H. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 1945–1950, Washington, DC, USA, 1999.

[5] A. Alfi, "PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems," *Acta Automatica Sinica*, vol. 37, no. 5, pp. 541–549, 2011.

[6] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural exponential inertia weight strategy in particle swarm optimization," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, pp. 3672–3675, Dalian, China, June 2006.

[7] Y.-L. Gao and Y.-H. Duan, "A new particle swarm optimization algorithm with random inertia weight and evolution strategy," in *Proceedings of the International Conference on Computational Intelligence and Security Workshops (CISW '07)*, pp. 199–203, Heilongjiang, China, December 2007.

[8] Y. Gao, X. An, and J. Liu, "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation," in *Proceedings of the International Conference on Computational Intelligence and Security (CIS '08)*, vol. 1, pp. 61–65, Suzhou, China, December 2008.

[9] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation (GECCO '09)*, pp. 1749–1750, Montreal, Canada, July 2009.

[10] H. R. Li and Y. L. Gao, "Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation," in *Proceedings of the 2nd International Conference on Information and Computing Science (ICIC '09)*, pp. 66–69, Manchester, UK, May 2009.

[11] H. Liu, R. Su, Y. Gao, and R. Xu, "Improved particle swarm optimization using two novel parallel inertia weights," in *Proceedings of the 2nd IEEE International Conference on Intelligent Computing Technology and Automation (ICICTA '09)*, pp. 185–188, Hunan, China, October 2009.

[12] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, and R. Ngah, "New particle swarm optimizer with sigmoid increasing inertia weight," *International Journal of Computer Science and Security*, vol. 1, no. 2, p. 35, 2007.

[13] X. Shen, Z. Chi, J. Yang, C. Chen, and Z. Chi, "Particle swarm optimization with dynamic adaptive inertia weight," in *Proceedings of the IEEE International Conference on Challenges in Environmental Science and Computer Engineering (CESCE '10)*, pp. 287–290, Wuhan, China, March 2010.

[14] Z. Jian-Ru, Z. Guo-Li, and Z. Hua, "Hybrid linear and nonlinear weight particle swarm optimization algorithm," in *Proceedings*

of the International Conference on Machine Learning and Cybernetics (ICMLC '12), vol. 3, pp. 1237–1241, July 2012.

[15] P. Chauhan, K. Deep, and M. Pant, "Novel inertia weight strategies for particle swarm optimization," *Memetic Computing*, vol. 5, no. 3, pp. 229–251, 2013.

[16] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 101–106, Seoul, Republic of Korea, May 2001.

[17] M. A. Arasomwan and A. O. Adewumi, "An adaptive velocity particle swarm optimization for high-dimensional function optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '13)*, pp. 2352–2359, Mexico City, Mexico, June 2013.

[18] M. A. Arasomwan and A. O. Adewumi, "On adaptive chaotic inertia weights in particle swarm optimization," in *Proceedings of the 4th IEEE Symposium Series on Computational Intelligence (SSCI '13)*, pp. 72–79, Singapore, 2013.

[19] S. Luke, *Essentials of Metaheuristics, a Set of Undergraduate Lecture Notes*, version 1.2, 1st edition, 2011.

[20] Y. H. Shi and R. C. Eberhart, "Amodified particle swarm optimizer," in *Proceedings of the IEEE International Conferences on Evolutionary Computation*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[21] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[22] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, John Wiley & Sons, New York, NY, USA, 2007.

[23] N. Iwasaki, K. Yasuda, and G. Ueno, "Dynamic parameter tuning of particle swarm optimization," *IEEJ Transactions on Electrical and Electronic Engineering*, vol. 1, no. 3, pp. 353–363, 2006.

[24] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., vol. 1447, pp. 591–600, Springer, Berlin, Germany, 1998.

[25] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 94–100, Seoul, Republic of Korea, May 2001.

[26] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3658–3670, 2011.

[27] M. Molga and C. Smutnicki, "Test functions for optimization needs," 2005, http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf.

[28] A. M. Montaz, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *Journal of Global Optimization*, vol. 31, no. 4, pp. 635–672, 2005.

[29] S. Chetty and A. O. Adewumi, "Three new stochastic local search algorithms for continuous optimization problems," *Computational Optimization and Applications*, 2013.

[30] G. I. Evers, *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization [M.S. thesis]*, University of Texas-Pan American, Edinburg, Tex, USA, 2009.

[31] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained global optimization," *Optimization Methods and Software*, vol. 26, no. 6, pp. 945–970, 2011.

Improved Particle Swarm Optimizer with Dynamically Adjusted Search Space and Velocity Limits

Akugbe Martins ARASOMWAN and Aderemi Oluyinka ADEWUMI, *Member*

***Abstract*— This paper presents an improved particle swarm optimization technique for global optimization. Many variants of the algorithm have been proposed in literature. However, two major things characterize many of these variants namely, static particle search space and velocity limits which bound their flexibilities in obtaining optimal solutions for many optimization problems. Besides, despite some additional parameters like inertia weight and extra computations in these variants compared with the original algorithm, the premature convergence of the original particle swarm algorithm remains a challenge. This paper proposes an improved particle swarm optimization algorithm without inertia weight. The proposed algorithm dynamically adjusts the search space and velocity limits for the swarm in each iteration by simply picking the highest and lowest values among all the dimensions of the particles, calculates their absolute values and use the higher of the two values to define a new search range and velocity limits for the next iteration. The efficiency and performance of the proposed algorithm was shown using popular benchmark global optimization problems with low and high dimensions. Results obtained demonstrate better convergence speed and precision, stability, robustness with better global search ability when compared with six recent variants of the original algorithm.**

***Index Terms*— Global optimization, particle swarm optimization, evolutionary computation, search space limits, swarm Intelligence, velocity limits**

I. Introduction

Individuals, enterprises and governments meet varieties of problems from day to day for which they seek best possible solutions amidst limited resources. Many of these problems can be formulated as optimization problems. The Original Particle Swarm Optimization (OPSO) [3] is a popular nature-inspired technique that displays problem-solving capabilities for researchers to solve complex and challenging optimization problems. It is an evolutionary computation technique inspired by social behaviour of birds and fish schooling. The concept was brought into optimization in 1995 [3, 12]

This work was supported in part by the College of Agriculture, Engineering and Sciences, University of Kwazulu-Natal, South Africa.

A. M. ARASOMWAN is with School of Mathematics, Statistics, and Computer Science University of Kwazulu-Natal South Africa Private Bag X54001, Durban, 4000 (e-mail: accuratesteps@yahoo.com).

A. O. ADEWUMI is with School of Mathematics, Statistics, and Computer Science University of Kwazulu-Natal South Africa Private Bag X54001, Durban, 4000 (e-mail: adewumia@ukzn.ac.za).

Global optimization seeks to provide solutions to optimization problems which are often multi-modal and non-convex. These solutions could all be globally good or a mix of globally and locally good solutions. While global optimization algorithms such as OPSO are most naturally applied to the optimization of multimodal cost functions, they can optimize unimodal functions as well. However, OPSO is often characterized with the problem of premature convergence. The quest for ridding OPSO of this problem and make the algorithm more efficient has led to many of its variants which are recorded in literature [4, 6, 14, 23, 25, 18], with many encouraging successes as well as resounding superiorities compared with OPSO algorithm. These variants have additional parameter(s) or extra (complex) computational effort(s), which without doubt should give them an edge over OPSO. Two major parameters common among OPSO variants are inertia weight and velocity threshold. Inertia weight was introduced into OPSO by [23] and helps the algorithm to balance its global and local search abilities while the velocity threshold, which helps control particle from searching outside the solution search space, has been extensively used in experimental studies in [24]. Solution search space is delimited by the upper and lower limits of the decision variables.

In many of the OPSO variants, solution search space and velocity threshold are static throughout the execution of the algorithm [1, 7, 8, 15, 20, 22, 24]. This characteristic somewhat limits the flexibilities of these variants in the process of obtaining optimal solutions for many of the optimization problems. Also, the common problem of premature convergence associated with OPSO remains unsolved by very many of the existing its variants [9, 16, 22]. In cases where (near) optimal solutions are obtained, they are with low precision [1, 6, 14, 20]. In order to obtain optimal results with higher precision for optimization problems by OPSO and many of its variants, there are needs to allow the solution search space and velocity threshold to vary dynamically based on the state of the particles' dimensions. This will enable the algorithm to concentrate its searching on the sub-range dynamically defined during its executing instead of searching the entire search space all the time. It could also enable the algorithm escape premature convergence.

In this paper, efforts were made to improve the performance of OPSO in terms of convergence speed, global search ability and increased solution accuracy, without additional parameters or complex computational efforts. The improved OPSO

algorithm (IOPSO), which does not use inertia weight, dynamically adjusted the search space and velocity limits for the swarm in each iteration by simply picking the highest and lowest values among all the dimensions of the particles, calculates their absolute values and then use the higher of the two values to define a new search range and velocity limits for next iteration. Empirical results from experiments performed showed that IOPSO is very efficient compared with the variants adopted for comparisons.

In the sections that follow, the framework of the OPSO is considered in Section 2, the OPSO variants adopted for comparisons are reviewed in Section 3, the proposed IOPSO technique is described in section 4, results of numerical simulations are presented in Section 5 and Section 6 concludes the paper.

II. THE FRAMEWORK OF ORIGINAL PSO (OPSO)

OPSO is a popular member of swarm intelligence metaheuristic. It is population-based, stochastic, robust, problem-independent and self-adaptive optimization technique for solving simple and complex optimization problems. It has been successfully used to solve many difficult real-world optimization problems [10, 17, 19]. When the technique was initially introduced, it was implemented with few lines of codes using basic mathematical operations; no major adjustment was needed to adapt it to new problems; it was almost independent of the initialization of the swarm; the gradient, continuity or differentiability of the problem to work with was not needed and very few parameters regulate the behaviour and efficiency, were required to be tuned to obtain quality solutions. Implementing this technique requires that the positions and velocities of a number of particles (swarm) be randomly generated using upper and lower bounds on the design variable values, after which the particles are randomly distributed in the solution search space. In the course of operation, every particle works with two major information – its personal experience and reachable neighbours' experiences; these are used to determine its next move in the solution space. Besides, each particle is associated with a value determined by the objective function of the problem being optimized to measure their qualities. The technique maintains a single swarm of particles throughout its execution and adjusts their positions and velocities in each iteration based on new discoveries about the solution space. These operations are basic to the implementations of OPSO variants.

The solution search space of the optimization search space is often represented as *n*-dimensional space. Also, the position and velocity of each particle are represented as the vectors $X_i = (x_{i1}, \ldots, x_{in})$ and $V_i = (v_{i1}, \ldots, v_{in})$, respectively. When the particles move in the search space searching for optimum solution to the problem being optimized, their velocities and positions are updated according to (1) and (2).

$$V_i(t+1) = V_i(t) + coeff_1(P_i - X_i) + ceff_2(P_g - X_i) \quad (1)$$

$$X_i(t+1) = X(t) + V_i(t+1) \quad (2)$$

Where P_i and P_g are vectors representing the i^{th} particle personal best and swarm global best positions respectively; $coeff_1 = c_1 r_1$ and $coeff_2 = c_2 r_2$; c_1 and c_2 are acceleration factors known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of P_i and P_g ; r_1 and r_2 are random numbers between 0 and 1 and t is iteration index. A value of 2.0 is used for c_1 and c_2 respectively.

The positions of particles in the swarm when they are being updated are controlled to be within some specified bounds as shown in (3), where *minX* and *maxX* represent the lower and upper bounds of the particle's position. Because the particle velocity based on (1), without restriction, could grow and make the particle oscillates around an optimum, increase its distance to the optimum on each iteration, or go out of the search space, the idea of velocity clamping was introduced by Eberhart and Kennedy in 1995 [5], into PSO to avoid the phenomenon of "swarm explosion". With this introduction, the particles could take reasonably sized steps so as to rake through the search space rather than bouncing about excessively. This has led to significant improvement as regards the performance of PSO. However, efforts have been made in time past to eliminate the use of velocity clamping, but researches have shown that velocity clamping has become a standard feature of OPSO [5]. Equation (4) shows one of the ways velocity clamping is implemented, with *minV* and *maxV* representing the lower and upper bounds of the particle's velocity.

$$x_i = \begin{cases} minX & if\ x_i < minX \\ maxX & if\ x_i > maxX \end{cases} \quad (3)$$

$$v_i = \begin{cases} minV & if\ x_i < minV \\ maxV & if\ x_i > maxV \end{cases} \quad (4)$$

OPSO being a stochastic population-based technique that relies directly on the objective values rather than the derivative information of the problem being optimized is less exposed to deception in the solution search space. However, it is susceptible to premature convergence, especially when the problem to be optimized is multi-peaked and when there are many decision variables (dimensions). This is because the more the particles communicate among themselves, the more they be alike until converging to the same region of the solution search space. If after some time no better global best is found by any other particle, they all converge about the existing global best which may not be the global minimizer.

III. OPSO VARIANTS ADOPTED FOR COMPARISONS

The OPSO variants considered for comparison with the proposed improved original PSO (IOPSO) in this paper are subsequently reviewed. These variants are AIW-PSO, iPSO, MARPSO, AIWPSO, PSO_{rank} and mPSO. All these variants implements (5) to update the velocities of particles, except otherwise clearly stated. Equation (5) differ from (1) because of the inertia weight parameter (ω) introduced into it. This parameter has attracted a lot of attentions and seems to be the most important compared with other parameters. The motivation behind its introduction was the desire to better control (or balance) the scope of the (local and global) search of OPSO algorithm and reduce the importance of (or

eliminate) velocity clamping, V_{max} during the optimization process [21, 23].

$$V_i(t+1) = \omega V_i(t) + coeff_1(P_i - X_i) + ceff_2(P_g - X_i) \quad (5)$$

A. Adaptive inertia weight PSO algorithm (AIW-PSO)

This variant was proposed in [22] to improve on balancing the global exploration and local exploitation abilities for PSO by taking advantage of the effect of inertia weight to achieve better results. A measure called individual search ability (ISA) defined in (6) was used to ascertain the current situation of each particle, i.e. whether the particle lacks global exploration or local exploitation abilities in each dimension. A large ISA means strong global exploration ability, inertia weight should be decreased. While a small ISA means that the inertia weight should be increased. This enables a particle decide whether to increase or decrease its values of inertia weight.

$$ISA_{ij} = \frac{|x_{ij} - p_{ij}|}{|p_{ij} - p_{gj}| + \varepsilon} \quad (6)$$

where, x_{ij} is the position of the i^{th} particle in the j^{th} dimension, p_{ij} is the own best solution, p_{gj} is the current global best solution, |…| denotes absolute value and ε is a positive constant close enough to zero.

Depending on the ISA, the inertia weight of i^{th} particle in j^{th} dimension was dynamically calculated in each iteration using a transform function defined in (7), so as to enhance the corresponding weak search abilities. This strategy was found to improve the performance of PSO algorithm.

$$\omega_{ij} = 1 - \alpha\left(\frac{1}{1 + e^{-ISA_{ij}}}\right) \quad (7)$$

where, α is a positive constant in the range (0,1].

B. Improved PSO (iPSO)

This variant used opposition-based learning to enhance the performance of PSO. The underlying principle behind this approach is the basic idea of opposition-based learning [15]: assuming a worst case, particle with the lowest fitness, $\boldsymbol{x}_b$, is taken to be a guess that is "very far away from the existing solution" in the opposite location. In each iteration this particle is replaced with its opposite (the anti-particle) as shown in (8).

$$x_{bj} = LB_j + UB_j - x_{bj} \quad (8)$$

where $x_{bj} \in [LB_j, UB_j]$, $j = 1, 2, \ldots, N_d$ and N_d is the dimension of the problem. LB_j and UB_j are the lower and upper bounds for the decision variable x, in the d^{th} dimension.

During each iteration, the velocity and personal experience of the anti-particle are reset while the global best solution is also updated.

C. Modified attractive and repulsive PSO (MARPSO)

MARPSO is a new diversity-guided PSO and a modification of the attractive and repulsive PSO (ARPSO) [8]. The major goal of this variant was to solve the problem of premature convergence associated with PSO by increasing the diversity of swarm, while maintaining a higher convergence speed. In achieving their goal, the authors introduced new measure of population diversity function and concept of the particle′s best flight direction into ARPSO. Because the algorithm could not guarantee local and global convergences, a mutation strategy was also introduced into it in order to improve its convergence. The algorithm used (9) to update the velocities of particles and maintained (2) for the particles' positions updating.

$$\begin{aligned} v_{ij}(t+1) = \omega v_{ij}(t) d_{ij}(t+1) + dir(t+1) \\ \times \left(c_1 r_{1j}\left(p_{ij} - x_{ij}(t)\right)\right. \\ \left. + c_2 r_{2j}\left(p_{gj} - x_{ij}(t)\right)\right) \end{aligned} \quad (9)$$

Where *dir(t)* as defined in (10) is the flight direction of the t^{th} generation and $d_i(t)$ in (11) is the flight direction of the i^{th} particle of the t^{th} generation.

$$dir(t+1) = \begin{cases} -1, & if\ (dir(t) > 0) \text{and} (diversity < d_{low}) \\ 1, & if\ (dir(t) < 0) \text{and} (diversity > d_{high}) \\ dir(t), & otherwise \end{cases} \quad (10)$$

The expression *dir(t)* = 1 means that the swarm does attractive movement while *dir(t)* = -1 means it does repulsive movement. The d_{low} and d_{high} are low and high limits of the particles respectively.

$$d_i(t+1) = \begin{cases} \dfrac{v(t)}{|v(t)|}, & if\ (f(x(t)) < f(p_i)) \\ d_i(t), & otherwise \end{cases} \quad (11)$$

The inertia part of (9) is beneficial to the search when $d_i(t)$ is 1 or -1. The diversity of the swarm represented by *diversity* is measured according to (12).

$$diversity = \frac{1}{sL} \times \sum_{i=1}^{s} \sqrt{\sum_{j=1}^{d} \left(P_{ij} - \bar{P}_j\right)^2} \quad (12)$$

The mutation strategy as used in the algorithm is defined in (13) for velocities of particles and (14) for the positions of particles.

$$v(t+1) = \begin{cases} V_{max} \times r_3^t, & if\ (|v(t)| < V_{min}) \text{and} (r_4 < 0.5) \\ -V_{max} \times r_3^t, & if\ (|v(t)| < V_{min}) \text{and} (r_4 \geq 0.5) \end{cases} \quad (13)$$

$$x(t+1) = \begin{cases} p_g + r_3^t, & if\ (|v(t)| < V_{min}) \text{and} (r_4 < 0.5) \\ p_g + r_3^t, & if\ (|v(t)| < V_{min}) \text{and} (r_4 \geq 0.5) \end{cases} \quad (14)$$

Where V_{min} and V_{max} are the low and high limits of the speed of particles while r_3, $r_4 \in U[0,1]$. It is evident in (13) and (14) that the mutation is carried out when the speed of the particle is less than V_{min}.

D. Adaptive inertia weight PSO (AIWPSO)

In [20], AIWPSO was proposed to further improve on the performance of PSO by introducing inertia weight that uses the swarm success rate to compute inertia weight by mapping it to a range of maximum and minimum inertia weight values $[\omega_{max}, \omega_{min}]$ using a linear function shown in (15). Using this

adaptive inertia weight value, the algorithm is able to improve the performance of PSO in the static and dynamic environments. To improve exploration, at the end of each iteration of the algorithm, the worst particle is replaced by a mutated best particle. The mutation is done by adding a Gaussian noise with zero mean standard deviation to one of the randomly chosen dimension of the best particle to facilitate exploration. AIWPSO outperformed its competitors virtually in all the numerical tests performed [20]. The adaptive inertia weights help to provide a knowledge of situation of the swarm at each iteration. A high percentage of success indicates that the particles have converged to a point that is far from the optimum point and the entire swarm is slowly moving towards the optimum while a low percentage of success shows that the particles are oscillating around the optimum without much improvement.

$$\omega_t = (\omega_{start} - \omega_{end})SP_t + \omega_{end} \quad (15)$$

where, ω_{start} and ω_{end} are predefined constants representing the initial and final values of the inertia weight. The success percentage in the t^{th} iteration ($SP_t \in [0,1]$) of the swarm is computed according to (16).

$$SP_t = \frac{\sum_i^n succ_t^i}{n} \quad (16)$$

where, n is the swarm size and the success of particle i in the t^{th} iteration ($succ_t^i$) is obtained using (17), with the assumption that a minimization problem is being considered.

$$succ_t^i = \begin{cases} 1 & f(Pbest_t^i) < f(Pbest_{t-1}^i) \\ 0 & f(Pbest_t^i) \geq f(Pbest_{t-1}^i) \end{cases} \quad (17)$$

where $Pbest_t^i$ is the current best position of particle i until iteration t and $f()$ is the function to be optimized.

E. Rank based PSO with dynamic adaptation (PSO_{rank})

In [1], a variation on the standard PSO algorithm called PSO_{rank} was proposed based on cooperative behavior of particles in the swarm. It uses a time-varying inertia weight which decreases non-linearly to improve its performance. In the algorithm, some of the best particles (which decrease in number as the iteration increases) are selected proportionate to their respective strengths, after the particles are ranked based on their fitness, so that they contribute to the updating of the position of a candidate particle. The strength of each contributing particle is a function of strivness, immediacy and number of contributed particles. The local search and convergence to global optimum solution by the algorithm depends on these selected best particles. PSO_{rank} updates the velocity vector of the particles using (18).

$$v_d^i(k+1) = \omega v_d^i(k) + rand_1\left(p_d^i - x_d^i(d)\right) + rand_2\left(\sum_{j=1}^{n} \psi_i^j \left(p_d^i(k) - x_d^i(d)\right)\right) \quad (18)$$

where, $\psi_i^j(k) = f\left(\tau_i^j(k), \delta_i^j(k), \xi_i\right) = \tau_i^j(k) \times \delta_i^j(k) \times \xi_i$ models the influence of the neighbour particle j on the candidate particle i in the k^{th} iteration, $\tau_i^j(k) = fitness_j(k) / \sum_{l=0}^{Neighbours_i} fitness_l(k)$ is the ranking parameter which signifies the strivness of the individual j in the neighbourhood of the i^{th} particle; $fitness_j(k)$ is the fitness of particle j in the neighbourhood of particle i and $Neighbours_i$ is the number of neighbour particles. $\delta_i^j(k) = 1/\sqrt{\sum_{d=1}^{D}(x_d^j(k) - x_d^i(k))^2}$ is the immediacy of individual j from particle i based on Euclidean distance in D-dimensional solution space where $x_d^j(k)$ and $x_d^i(k)$ respectively represent the positions of the particle j and the candidate particle i in dimension d of the solution space. $\xi_i = \alpha N_i^{\beta}$ is the effect of the individuals in the neighbourhood of the i^{th} particle, where N_i is the number of individuals in the neighbourhood of particle i; $0 < \alpha < 1$ and $0 < \beta < 1$ are parameters which controls the importance of social knowledge provided by the neighbour individuals.

F. Modified PSO (mPSO)

This variant [7], addressed the issue of particles getting over concentrated, tried to delay the algorithm falling into local minimum and increase the global search capability of the swarm. The authors used (19) to control the swarm diversity effectively in order to prevent their quick gathering at the location of *gbest*. This was done with the belief that, effective control of the swarm's aggregation degree will improve the algorithm's capability to obtain global minimum.

$$newPbest_d = Pbest_d \times (1 + \eta\sigma) \quad (19)$$

From (19), σ is a random number drawn from the standard Gaussian distribution, the initial value of the $\eta = 1.0$, and set $\eta = \beta\eta$ every 50 iterations, where β is a random number between [0.01, 0.9]. This method not only produces a small range of disturbance to achieve the local search with high probability, but also produces a significant disturbance to step out of the local minimum area with large step migration in time.

G. Discussions

All the variants described above tried to address the problem of getting stuck in local optima (premature convergence) common with OPSO. In the process of trying to achieve their goals, the authors of these variants modified OPSO in various ways by introducing additional parameters and computations. All the variants outperformed their competitors in solving various test problems that were used in the different experiments conducted by their authors. Summarized in Table I are the additional parameters to OPSO and the extra computations associated with these variants.

TABLE I: ADDITIONAL PARAMETERS AND COMPUTATIONS IN THE COMPETING OPSO VARIANTS

No.	Variant	Additional parameters and computations	Remark
1	AIW-PSO	i. ISA ii. V_{max} and V_{min} iii. ω	(i) and (iii) were computed. (ii) was assigned the

			search space limits
2	iPSO	i. ω	Was set as a constant
3	MARPSO	i. Flight directions (for the swarm and each particle ii. Diversity iii. Mutation of particles' positions iv. ω v. d_{high} and d_{low} vi. V_{max} and V_{min}	(i) – (v) were computed and no value was explicitly assigned to (vi)
4	AIWPSO	i. SR ii. Mutation of selected particle iii. ω iv. V_{max} and V_{min}	(i) – (iii) were computed while (iv) was not explicitly stated whether it was assigned a constant value or the search space threshold
5	PSO_{rank}	i. Ranking of particles ii. Influence iii. Euclidean distance iv. Individual effects v. β vi. α vii. V_{max} and V_{min} viii. ω	(i) – (iv) and (viii) were computed. (v) and (vi) were set experimentally while (vii) was assigned the search space threshold
6	mPSO	i. σ ii. η iii. newPbest iv. V_{max} and V_{min} v. ω	(i), (ii), (iii) and (v) were computed while (iv) was assigned a constant value

Considering the OPSO variants described above, MARPSO and PSO_{rank} are more complicated than others while *i*PSO is the least complicated in terms of extra computations and additional parameters. In this regard, *i*PSO strived at maintaining the goal of being a simple algorithm which was one the desires of the authors of OPSO [12], but could not maintain the goal of robustness. For mPSO, AIWPSO, MARPSO and PSO_{rank}, the goal of robustness was achieved to a very high level, but could not maintain the goal of simplicity. From Table I, inertia weight (ω) and particles velocity limits (V_{max} and V_{min}) parameters are common among the variants. In cases where the velocity limits were assigned the upper and lower limits of the solution search space of a problem, the values remained constant throughout the lifetime of the algorithm [1, 22]. This was equally the same thing when the velocity threshold was assigned constant values relative to each problem [7]. Also, in all the variants as it is common among other OPSO variants, the solution search space remains constant till the algorithms finish their executions.

The inertia weight and velocity threshold plays important roles in the exploration and exploitation ability of PSO algorithm, though their selections may be problem-dependent. There are possibilities of encountering some practical problems with lack of knowledge regarding the selection of V_{max} which could result to using trial-and-error approach in order to make a selection and this can be very labourious and time consuming. Allowing the velocity threshold to remain static, either by assigning to it a predefined constant value or a search space threshold, throughout the lifetime of the algorithms can make the particles have some step size that may make them do more than enough exploration or less than enough exploitation. The inertia weight parameter is the common tool being used to address this challenge, but this could better be addressed by working directly with the velocities of the particles because it is the direct determinant of the particles' step sizes. Making the solution search spaces static could also make the particles spend needless time searching areas that may not be necessary for solution. If the velocity and solution search space limits are made to vary (dynamic) throughout the lifetime of the algorithms without using the inertia weight parameter, there are possibilities of obtaining better and quality solutions to optimization problems. This is what the present paper seeks to achieve.

IV. THE IMPROVED ORIGINAL PSO (IOPSO)

All the PSO variants considered in this paper obtained solutions for the test problems that were used to validate the, with varied solution quality and precision. These variants have additional parameters and some extra (or complex) computations that enabled them achieve their various levels of successes. The major goal of this paper is to improve on the performance of OPSO, in a simple way, without using the inertia weight parameter (ω) or getting involved in complex computation(s). Apart from the commonly used V_{max} and velocity clamping percentage (represented as μ in this paper), no other parameters were used. This was done to make the algorithm simple and robust yet very effective.

In order to achieve this major goal, a careful study was done regarding the particles' dimensions. First, the following observations were made:

i. During search, every particle dynamically changes its position in a complex environment facing different situation. As a result, each particle along every dimension may have different trade-off between global and local search abilities
ii. Clamping the velocity of a particle changes the step size as well as the particle's direction since changing any component of a vector changes that vector's direction. As each dimension is optimized independently, the particle moves toward the global best on each dimension with a speed depending on the velocity limits. Since the maximum iterative movement toward global best on any dimension is clamped, particles may be thought of as combing the search space a bit more thoroughly than when their velocities are unclamped [5]

iii. It has also been experimentally discovered that large velocity threshold enhances exploration while small velocity threshold enhances exploitation [24]
iv. A minimizer is sought for the optimization problem
v. The final fitness (objective function) value depends on the values of the various dimensions that make up the minimizer
vi. When the algorithm terminates, the final values at the various dimensions of the minimizer are smaller than their initial values (when they were initialized at the beginning of the algorithm)
vii. From the foregoing, the primary purpose of an optimization algorithm is to optimize the values of each dimension (decision variables) from its initial value to a smaller (final) value such that the objective function value is the possible minimum (i.e. for minimization problems).

Second, an experimental study was conducted using OPSO to observe the progressive values of the dimensions as well as the fitness value for each particle while the algorithm is being run. The *Ackley* problem was used for the experiment, with dimension of 10, swarm size of 10, upper and lower particle velocity limits set as $v_{max} = x_{max}$ and $v_{min} = x_{min}$ and a maximum iteration of 100. Sample results for the values of the different dimension and fitness at the initialization state, as well as at the 10th, 20th, 50th and 100th iteration relative to the particles are shown in Appendix 1. From the results, it was discovered that algorithm

When the velocity and search space limits were allowed to vary dynamically (method described below), the experiment was repeated with the same settings. It was discovered that there was a great improvement as shown by the sample results in Appendix 2. The velocity threshold was used to control exploitation while the search space threshold was used to control exploration.

In every iteration, the largest dimension value (L_d) and the smallest dimension value (S_d) among the dimensions of all the particles, were obtained according to (20) and (21).

$$L_d \leftarrow \max_i \left(\max_j \left(x_i^j \right) \right) \tag{20}$$

$$S_d \leftarrow \min_i \left(\min_j \left(x_i^j \right) \right) \tag{21}$$

where, x_i^j is the i^{th} particle with j^{th} dimension. The upper limit x_{max} and lower limit x_{min} of the solution search space for the particles were obtained according to (22) and (23).

$$x_{max} \leftarrow max(|L_d|, |S_d|) \tag{22}$$

$$x_{min} \leftarrow -x_{max} \tag{23}$$

where $|\,.\,|$ means absolute value. After obtaining x_{max} and x_{min}, they are used to compute the upper (v_{max}) and lower (v_{min}) particle velocity limits as defined in (24) and (25).

$$v_{max} \leftarrow \mu x_{max} \tag{24}$$

$$v_{min} \leftarrow \mu x_{min} \tag{25}$$

where, μ is a velocity clamping percentage. It serves as a scaling factor of the upper and lower solution space limits to help reduce the velocity range for particles in the process of operation by IOPSO.

After obtaining the new velocity limits and solution search space, the particles are redistributed in the search space. When the particles' positions are being updated, contrary to the common method in (3) for ensuring that the particles do not move out of the solution search space, IOPSO uses Fig. 1. This method in some way help the algorithm achieve some level of exploration.

```
If x_i < x_min
  x_i ← x_min + (x_min − x_i)* random(0,1)
else if x_i > x_max
  x_i ← x_max − (x_i − x_max)* random(0,1)
end if
```

Fig. 1: Particle position clamping

Shown in Fig. 2 below is the algorithm for IOPSO. The shaded portions indicate areas of improvements made to OPSO.

1) Exploration feature of IOPSO

In order to be able to leave a current peak and look for better solutions in the search space, IOPSO utilizes (20) – (23) to redistribute the particles within the newly calculated solution search space. This method could provide the particles with the opportunity of leaving their current positions to other parts of the search space, thus helping to escape getting stuck in local optimum. This happens throughout the process of the algorithm.

2) Exploitation feature of IOPSO

To facilitate the refinement of the best solution it has found so far, (24) and (25) enable IOPSO to search a small vicinity of this solution. This is so because, as the algorithm's operation progresses, the velocity range of the particles decreases, thereby reducing the distance each particle should exploit for better solution and the smaller the velocity range the higher the exploitation by the particles.

V. NUMERICAL SIMULATIONS

To validate the performance of the proposed IOPSO, a total of 6 different recent and efficient PSO variants, namely: AIW-PSO, iPSO, MARPSO, AIWPSO, mPSO and PSO_{rank} were adopted for comparisons. Different experiments, relative to the competing PSO variants, used different set of test problems which were also used to test IOPSO. The application software was developed in Microsoft Visual C# programming language.

```
Begin IOPSO Algorithm
   Input:   f: the function to optimize
    s: the swarm size
    d: the problem dimension
    pr: solution search space
    vr: particle velocity range
   Output:  x*: the best particle position found
    f*: the best fitness value found
   Initialize: position x_i = (x_i1, …, x_id) and velocity v_i = (v_i1, …, v_id),
               for all particles in problem space
   evaluate f(x_i) in d variables and get pbest_i, (i = 1, …, s)
   gbest ← best of pbest_i
   While stopping criteria is false do
      compute new solution search range, pr, using  (20) – (23)
      compute new particle velocity range, vr, using  (24) and (25)
      randomly redistribute particles in the new pr
      randomly reinitialize velocities for particles using the new vr
      Loop for s times
       Loop for d times
           update v_i for particle using (1)
           validate for velocity boundaries using (4)
           update x_i for particle using (2)
           validate for position boundaries using Fig. 1
       End
      End
      If f(x_i) < f(pbest_i) then pbest_i ← x_i
      If f(x_i) < f(gbest) then
           gbest ← x_i
           f(gbest) ← f(x_i)
      end if
   End while
   x* ← gbest
   f* ← f(gbest)
   Return x* and f*
End IOPSO Algorithm
```

Fig. 2: Algorithm for IOPSO

A. Benchmark problems

Different test problems (see Table III) with varied difficulties that are diverse enough to cover many of the problems which can arise in global optimization problems were used to verify the performance of IOPSO, in comparison with the competing variants. All the test problems were obtained from [1, 7, 8, 15, 20, 22]. Some of the characteristics (US – unimodal separable, UN – unimodal non-separable, MS – multimodal separable, MN – multimodal non-separable) were obtained from [11].

B. Parameter setting

Two parameters were set for IOPSO, irrespective of their values used in the various competing variants. These parameters are c_1, c_2, andμ. Different values like 0.5, 0.15, 0.05, 0.75 and 1.0 have been used for μ in the literatures [1, 2, 5, 13, 22]. In this paper, the value for μ in (24) and (25) is 0.15; this value was used because it has been proved to be good and efficient [5, 13]. The value for c_1 and c_2 was 2.0; this value has also been proved to be generally good and are commonly used in the literature [1, 2, 8, 23]. The parameters r_1 and r_2 were randomly generated using the uniform random number generator. Inertia weight parameter ω was not used in IOPSO.

C. Settings of the experiment

A total of 6 different experiments were conducted. The settings of each experiment were relative to the competing variant as recorded in the respective literature. The settings for all the experiments are stated below one after the other, relative to each competing variant. Table II summarizes the different testing ground to prove the robustness, convergence speed, solution quality and stability of the algorithms. The initial positions of particles in IOPSO were generated using uniform random number generator.

TABLE II: SETTINGS FOR PSO VARIANTS ADOPTED FOR COMPARISONS WITH IOPSO

No.	Variant	Testing ground			
		Particles initialization	Swam sizes	Problem dimensions	Number of test problems
1	AIW-PSO	Asymmetric	20, 40 and 80	10, 20 and 30	3
2	iPSO	Symmetric	50	30	7
3	MARPSO	Symmetric	20	20, 50 and 100	4
4	AIWPSO	Symmetric	20	2 and 30	12
5	PSO_{rank}	Asymmetric	30	2, 10, 20 and 30	6
6	mPSO	Symmetric	30	2, 10, 20 and 30	6

TABLE III: BENCHMARK PROBLEMS

No.	Problem	Formulation	Characteristics	Optimum
1	Ackley	$f(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	MN	0
2	Griewank	$f(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	MN	0
3	Noisy Quadric	$f(\vec{x}) = \sum_{i=1}^{d} i x_i^4 + random(0,1)$	US	0
4	Noncontinous Rastrigin	$f(\vec{x}) = \sum_{i=1}^{d} (y_i^2 - 10\cos(2\pi y_i) + 10)$ $y_i = \begin{Bmatrix} x_i & if\ \lvert x_i\rvert < 0.5 \\ \frac{round(2x_i)}{2} & if\ \lvert x_i\rvert \geq 0.5 \end{Bmatrix}$	MS	0
5	Rastrigin	$f(\vec{x}) = \sum_{i=1}^{d} (x_i^2 - 10\cos(2\pi x_i) + 10)$	MS	0
6	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2) + (x_i - 1)^2$	UN	0
7	Rotated Ellipsoid	$f(\vec{x}) = \sum_{i=1}^{d} \left(\sum_{j=1}^{i} x_j\right)^2$	UN	0
8	Salomon	$f(\vec{x}) = -cos\left(2\pi\sum_{i=1}^{d} x_i^2\right) + 0.1\sqrt{\sum_{i=1}^{d} x_i^2} + 1$	MN	0
9	Schaffer's f6	$f(\vec{x}) = \sum_{i=1}^{d-1} \left(0.5 + \frac{sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2}\right)$	MN	0
10	Schwefel P2.22	$f(\vec{x}) = \sum_{i=1}^{d} \lvert x_i\rvert + \prod_{i=1}^{d} \lvert x_i\rvert$	UN	0
11	Shubert	$f(\vec{x}) = \prod_{i=1}^{d} \left(\sum_{j=1}^{5} jcos((j+1)x_i + j)\right)$	MN	-186.7309
12	Sphere	$f(\vec{x}) = \sum_{i=1}^{d} x_i^2$	US	0
13	Step	$f(\vec{x}) = \sum_{i=1}^{d} (\lfloor x_i + 0.5\rfloor)^2$	US	0

1) Experiment 1

In this experiment IOPSO was compared with AIW-PSO [22]. The test problems, search ranges and initialization ranges are stated in Table IV while their dimensions and swarm sizes in Table II. The maximum numbers of iterations was set to 1000, 1500 and 2000 respectively for the dimensions. The experiment was repeated 100 times for each test problem.

TABLE IV: SETTINGS FOR EXPERIMENT 1

Test problem	Griewank	Rastrigin	Rosenbrock
Search range	[-600,600]	[-10,10]	[-100,100]
Initialization range	[300,600]	[2.56,5.12]	[15,30]

2) Experiment 2

In this experiment IOPSO was compared with *i*PSO [15]. The test problems dimensions and swarm sizes are stated in Table II. The total number of simulations was 30 and each simulation was allowed to run for 50,000 evaluations of the objective function. Shown in Table IV are test problems, search ranges and initialization ranges.

3) Experiment 3

In this experiment IOPSO was compared with MARPSO [8]. The test problems, search ranges and initialization ranges are stated in Table VI while their dimensions and swarm sizes are stated in Table II. The number of evaluations of objective function for the different problem dimensions was set to 40000, 100000, and 200000 respectively. Error tolerance was set to 10^{-10}, that is, fitness smaller than error tolerance was considered as zero. For IOPSO, the experiment was repeated 100 times for each of the test problems.

4) Experiment 4

In this experiment IOPSO was compared with AIWPSO [20]. The test problems, search ranges and initialization ranges are stated in Table VII while their dimensions and swarm sizes are stated in Table II. The maximum allowed number of function evaluations was set to 200,000. The experiment was repeated 30 times for each of the test problems.

5) Experiment 5

In this experiment IOPSO was compared with PSO_{rank} [1]. The test problems, search ranges and initialization ranges are stated in Table VII while their dimensions and swarm sizes are stated in Table II. The maximum numbers of iterations was set to 1000, 1500 and 2000 for 10, 20 and 30 dimensions respectively. For Schaffer's f6 problem, the maximum iteration was set to 1000. The experiment was repeated 100 times for each test problem. Success criterion was set for all the problems; for Schaffer's f6, success criterion was set to 10^{-6} and 10^{-2} for others. After the maximum iteration, if the minimum value reached by the algorithm was not below the threshold, the run was considered unsuccessful. Average fitness smaller than 10^{-15} was considered as zero.

6) Experiment 6

In this experiment IOPSO was compared with mPSO [7]. The test problems, search ranges and initialization ranges are stated in Table IX while their dimensions and swarm sizes are stated in Table II. The maximum numbers of iterations was set to 3000 for all dimensions. The experiment was repeated 50 times for each test problem. The target value of function optimization was set to 10^{-10}. After the maximum iteration, any average fitness smaller than 10^{-10} was considered to be zero. The goal of this experiment was to verify whether IOPSO.

TABLE V: SETTINGS FOR EXPERIMENT 2

Test problem	Ackley	Griewank	Rastrigin	Rosenbrock	Rotated hyper-ellipsoid	Salomon	Sphere
Search range	[-32,32]	[-600,600]	[-5.12,5.12]	[-2,2]	[-100,100]	[-100,100]	[-100,100]
Initialization range	[-32,32]	[-600,600]	[-5.12,5.12]	[-2,2]	[-100,100]	[-100,100]	[-100,100]

TABLE VI: SETTINGS FOR EXPERIMENT 3

Test problem	Ackley	Griewank	Rastrigin	Rosenbrock
Search range	[-32,32]	[-600,600]	[-5.12,5.12]	[-30,30]
Initialization range	[-32,32]	[-600,600]	[-5.12,5.12]	[-30,30]

TABLE VII: SETTINGS FOR EXPERIMENT 4

Test problem	Ackley	Griewank	Noisy Quadric	NC Rastrigin	Rastrigin	Rosenbrock
Search range	[-32,32]	[-600,600]	[-1.28,1.28]	[-30,30]	[-5.12,5.12]	[-5,10]
Initialization range	[-32,32]	[-600,600]	[-1.28,1.28]	[-30,30]	[-5.12,5.12]	[-5,10]

Test problem	Rotated-ellipsoid	Schwefel	Schwefel P2.22	Shubert	Sphere	Step
Search range	[-100,100]	[-500,500]	[-10,10]	[-10,10]	[-100,100]	[-100,100]
Initialization range	[-100,100]	[-500,500]	[-10,10]	[-10,10]	[-100,100]	[-100,100]

TABLE VIII: SETTINGS FOR EXPERIMENT 5

Test problem	Ackley	Griewank	Rastrigin	Rosenbrock	Schaffer's f6	Sphere
Search range	[-30,30]	[-600,600]	[-5.12,5.12]	[-30,30]	[-100,100]	[-100,100]
Initialization range	[15,30]	[300,600]	[2.56,5.12]	[15,30]	[50,100]	[50,100]

TABLE IX: SETTINGS FOR EXPERIMENT 6

Test problem	Ackley	Griewank	Rastrigin	Rosenbrock	Schaffer's f6	Sphere
Search range	[-30,30]	[-600,600]	[-5.12,5.12]	[-30,30]	[-5.12,5.12]	[-1000,1000]
Initialization range	[-30,30]	[-600,600]	[-5.12,5.12]	[-30,30]	[-5.12,5.12]	[-1000,1000]

D. Comparative Study and Discussions

Results obtained from all the experiments are presented and discussed in this sub-section to show the overall performance of the proposed method compared to other methods. In all the comparisons, mean best solution (Mean Fitness) is a measure of the precision that the algorithm can get within given iterations while standard deviation (Std. Dev.) is a measure of the algorithm's stability and robustness and success rate (SR) is the rate of the optimum fitness result in the criterion range experimenting a number times independently.

Recorded in Tables X – XV are the numerical results obtained for all the experiments. All the results for the competing PSO variants were obtained from the respective referenced papers and they are presented here the way they were recorded. Thus, the recording of the results for IOPSO were patterned after them. In each of the tables, for ease of observation bold values represent the best results.

1) Results for Experiment 1

The results in Table X clearly reveal a great difference in performance between IOPSO and AIW-PSO. The results are compared based on the final accuracy of the averaged best solutions. In all the test problems across the swarm sizes and dimensions except dimension 30 with swarm size 80, results indicate that IOPSO can get better optimum fitness results, showing better convergence precision better global search ability compared with AIW-PSO. The solution obtained for *Rastrigin* problem when the swarm size was 80 and dimension 30, shows that IOPSO was not comfortable working with large swarm size relative to the problem, under the limitation of allowed maximum number of iterations. IOPSO obtained optimal solutions for *Griewank* and Rastrigin, but it was trapped in local minimum solving *Rosenbrock*.

2) Results for Experiment 2

In Tables XI, IOPSO is compared together with *i*PSO based on their final accuracies of the averaged best solutions and stability. Both algorithms performed equally in *Ackley* and *Sphere* problems. IOPSO demonstrated better search ability to obtain optimal minimum with better accuracy and stability for *Griewank*, *Rotated Ellipsoid* and *Salomon*. For *Rosenbrock*, the two algorithms could not obtain optimal minimum but *i*PSO was better in solution accuracy while IOPSO was better in algorithm stability. *i*PSO performed better in *Rastrigin*; this is because IOPSO was not comfortable working with the large swarm size within the allowed number of iterations.

TABLE X: THE BEST FITNESS VALUES FOR IOPSO AND AIW-PSO

Swarm size	**Dimension**	**Max Iteration**	**Griewank**		**Rastrigin**		**Rosenbrock**	
			AIW-PSO	*IOPSO*	*AIW-PSO*	*IOPSO*	*AIW-PSO*	*IOPSO*
20	10	1000	0.0734	**0.0000**	3.7415	**0.0000**	48.6378	**8.9251**
	20	1500	0.0252	**0.0000**	11.1323	**0.0000**	115.1627	**18.9097**
	30	2000	0.0120	**0.0000**	22.1155	**0.0000**	218.9012	**28.9001**
40	10	1000	0.0671	**0.0000**	1.9900	**0.0000**	24.5149	**8.9160**
	20	1500	0.0266	**0.0000**	7.2145	**0.0000**	60.0686	**18.8963**
	30	2000	0.0146	**0.0000**	17.5765	**0.0000**	128.7677	**28.8835**
80	10	1000	0.0106	**0.0000**	1.0051	**0.0000**	19.2232	**8.9088**
	20	1500	0.0258	**0.0000**	5.0615	**2.0800**	52.8523	**18.8815**
	30	2000	0.0106	**0.0000**	**13.1237**	19.5370	149.4491	**28.8659**

TABLE XI: RESULTS FOR IOPSO AND *i*PSO FOR THE SCALABLE BENCHMARK PROBLEMS IN 10

Problem	Ackley		Griewank		Rastrigin		Rosenbrock	
Method	*iPSO*	*IOPSO*	*iPSO*	*IOPSO*	*iPSO*	*IOPSO*	*iPSO*	*IOPSO*
Mean Fitness	0.000000	0.000000	0.006163	**0.000000**	**27.460845**	29.578568	**20.645323**	28.879818
Std. Dev.	0.000000	0.000000	0.009966	**0.000000**	**11.966896**	56.553040	0.426212	**0.016367**

Problem	Rotated Ellipsoid		Salomon		Sphere	
Method	*iPSO*	*IOPSO*	*iPSO*	*IOPSO*	*iPSO*	*IOPSO*
Mean Fitness	0.355572	**0.000000**	0.113207	**0.099834**	0.000000	0.000000
Std. Dev.	0.890755	**0.000000**	0.034575	**0.000002**	0.000000	0.000000

3) Results for Experiment 3

Table XII compares the results of IOPSO with that of MARPSO based on their final accuracies of the averaged best solutions. IPSO generally performed better than MARPSO because it was able to obtain optimal minimum for 3 (*Ackley*, *Griewank* and *Rastrigin*) out of the 5 problems across the problems dimensions whereas MAPSO was able to obtain optimal minimum for *Rastrigin*. IOPSO was able to obtain better result for *Rastrigin* because a swarm size of 20 was used. For *Rosenbrock*, the two algorithms could not obtain optimal minimum but the solutions obtained by MARPSO across the problems dimensions were better in terms of accuracy.

4) Results for Experiment 4

Table XIII represents two measures (mean fitness and standard deviation) for the experimental results obtained by IOPSO and AIWPSO in 12 problems. Out of these problems, IOPSO significantly outperforms AIWPSO in 8 of them while AIWPSO obtained better results in 3 of them. But the two algorithms successfully optimized the *Step* problem with equal precision, quality and stability. IOPSO was able to obtain optimal minimum for *Rastrigin* and *Non-continuous Rastrigin (NC Rastrigin)* because the swarm size used for the experiment was 20. In *Griewank* and *Rastrigin* problems, while AIWPSO became trapped in the local optima, IOPSO was able to escape and obtained optimal results. Though AIWPSO got good results in *Ackley*, *Rotated Ellipsoid, Schwefel P2.22* and the problem with noise (*Noisy Quadric)*, but IOPSO excelled it by obtaining results with better quality, precision and stability. The two algorithms could not obtain optimal solution for *Rosenbrock* and *Schwefel*, but AIWPSO obtained better solutions for them than IOPSO; however, IOPSO was more stable optimizing *Rosenbrock*. AIWPSO obtained optimal solution for *Shubert* but IOPSO could not because its results for some of the runs were not optimal which affected its average performance in the problem. Considering the entire set of problems, IOPSO demonstrated better global search ability, convergence quality and speed, as well as stability compared with AIWPSO.

5) Results for experiment 5

The numerical results for the test problems are recorded in Tables XIV (a) – (d). The results are presented in order of problem dimensions for the scalable problems and then followed by the *Schaffer's f6* problem. The two algorithms successfully optimized *Rastrigin* across the three different problem dimensions and *Schaffer's f6* which is of dimension 2. They demonstrated equal stability, search ability and obtained solutions with same quality. In *Rosenbrock* problem, PSO_{rank} obtained solutions with better quality and demonstrated better search ability in all the dimensions, but IOPSO was more stable. Though the two algorithms had equal success rate in *Ackley, Griewank* and *Sphere* across the dimensions, IOPSO significantly outperformed PSO_{rank} in result quality, robustness, stability and global search ability.

6) Results for experiment 6

Tables XV(a) – (d) show all the results of IOPSO and *m*PSO for six problems. The two algorithms were tested with different dimensions of the problems and four measures as shown in the tables were used to verify their performances. From the results, the two algorithms could not reach the given target value when optimizing *Rosenbrock* problem; while *m*PSO obtained a result with little difference in accuracy, IOPSO was more stable across the dimensions. For the other problems, in terms of result quality, convergence accuracy, robustness, algorithm stability and global search ability the two algorithms had equal performance in optimizing the problems. However, the results revealed that the two algorithms differ in terms of convergence speed. Across the problem dimensions, IOPSO converged twice as fast as *m*PSO in *Ackley* and *Sphere* problems but almost twice as fast as *m*PSO in *Griewank* problem. Optimizing *Rastrigin* problem, IOPSO also converge faster than *m*PSO when the problem dimension was set to 10, but as the problem dimension was increased to 20 and 30, *m*PSO converged faster than IOPSO. In *Schaffer's f6* problem, *m*PSO had higher convergence speed than IOPSO.

TABLE XII: THE BEST FITNESS VALUES FOR IOPSO AND MARPSO

Dimension	Function evaluation	Ackley		Griewank		Rastrigin		Rosenbrock	
		MARPSO	*IOPSO*	*MARPSO*	*IOPSO*	*MARPSO*	*IOPSO*	*MARPSO*	*IOPSO*
20	40000	0.00e+00	0.00e+00	4.03e-03	**0.00e+00**	0.00e+00	0.00e+00	**0.13**	1.89e+01
50	100000	2.39e-10	**0.00e+00**	1.97e-04	**0.00e+00**	0.00e+00	0.00e+00	**1.28**	4.89e+01
100	200000	3.99e-09	**0.00e+00**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**16.93**	9.89e+01

TABLE XIII: RESULTS FOR IOPSO AND AIWPSO

Problem	Ackley		Griewank		Noisy Quadric		NC Rastrigin	
Method	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*
Mean Fitness	6.9870e-15	**4.4409e-16**	2.8524e-02	**0.0000e+00**	5.5241e-03	**6.0166e-06**	1.1842e-16	**0.0000e+00**
Std. Dev.	4.2073e-31	**0.0000e+00**	7.6640e-04	**0.0000e+00**	1.5358e-05	**6.0163e-06**	4.2073e-31	**0.0000e+00**

Problem	Rastrigin		Rosenbrock		Rotated Ellipsoid		Schwefel	
Method	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*
Mean Fitness	1.6583e-01	**0.0000e+00**	**2.5003e+00**	2.8901e+01	1.9570e-10	**4.2366e-256**	**-1.1732e+04**	-2.7375e+03
Std. Dev.	2.1051e-01	**0.0000e+00**	1.5978e+01	**1.9131e-02**	1.2012e-19	**0.0000e+000**	**1.1409e-25**	4.2861e+02

Problem	Schwefel P2.22		Shubert		Sphere		Step	
Method	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*	*AIWPSO*	*IOPSO*
Mean Fitness	1.6534e-62	**2.2083e-206**	**-1.8673e+02**	-1.7717e+02	3.3703e-134	**0.0000e+00**	0.0000e+00	0.0000e+00
Std. Dev.	7.7348e-123	**0.0000e+000**	**1.0306e-27**	9.8384e+00	5.1722e-267	**0.0000e+00**	0.0000e+00	0.0000e+00

TABLE XIV(A): RESULTS FOR IOPSO AND PSO_{RANK} FOR THE SCALABLE BENCHMARK WITH DIMENSION OF 10

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*
Mean Fitness	1.31e-06	**0.00e+00**	2.53e-05	**0.00e+00**	0.00e+00	0.00e+00	**9.14e-03**	8.92e+00	1.21e-10	**0.00e+00**
Std. Dev.	6.54e-06	**0.00e+00**	3.47e-o5	**0.00e+00**	0.00e+00	0.00e+00	1.42e-02	**7.31e-03**	8.36e-10	**0.00e+00**
SR	1	1	1	1	1	1	0.96	0	1	1

TABLE XIV(B): RESULTS FOR IOPSO AND PSO_{RANK} FOR THE SCALABLE BENCHMARK WITH DIMENSION OF 20

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*PSO_{rank}*	*AIWPSO*	*PSO_{rank}*
Mean Fitness	4.22e-06	**0.00e+00**	4.47e-07	**0.00e+00**	0.00e+00	0.00e+00	**1.61e+00**	1.89e+01	1.08e-09	**0.00e+00**
Std. Dev.	9.11e-06	**0.00e+00**	7.69e-07	**0.00e+00**	0.00e+00	0.00e+00	2.04e+00	**1.46e-02**	3.76e-09	**0.00e+00**
SR	1	1	1	1	1	1	0.56	0	1	1

TABLE XIV(C): RESULTS FOR IOPSO AND PSO_{RANK} FOR THE SCALABLE BENCHMARK WITH DIMENSION OF 30

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*	*PSO_{rank}*	*IOPSO*
Mean Fitness	3.12e-05	**0.00e+00**	2.73e-08	**0.00e+00**	0.00e+00	0.00e+00	**1.27e+01**	2.89e+01	2.05e-08	**0.00e+00**
Std. Dev.	8.35e-05	**0.00e+00**	5.24e-08	**0.00e+00**	0.00e+00	0.00e+00	1.39e+01	**1.52e-02**	6.41e-08	**0.00e+00**
SR	1	1	1	1	1	1	0.19	0	1	1

TABLE XIV(D): RESULTS FOR IOPSO AND PSO_{RANK} FOR SCHAFFER'S F6

Problem	Schaffer's f6	
Method	*PSO_{rank}*	*IOPSO*
Mean Fitness	0.00e+00	0.00e+00
Std. Dev.	0.00e+00	0.00e+00
SR	1	1

TABLE XV(A): RESULTS FOR IOPSO AND MPSO FOR THE BENCHMARK WITH DIMENSION OF 10

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*
Mean Fitness	0.000	0.000	0.000	0.000	0.000	0.000	**8.253**	8.915	0.000	0.000
Std. Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.210	**0.009**	0.000	0.000
Avg. Iteration	468.08	**203.82**	294.32	**176.42**	315.24	**239.92**	3000	3000	340.18	**139.12**
SR	50/50	50/50	50/50	50/50	50/50	50/50	0/50	0/50	50/50	50/50

TABLE XV(B): RESULTS FOR IOPSO AND MPSO FOR THE BENCHMARK WITH DIMENSION OF 20

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*
Mean Fitness	0.000	0.000	0.000	0.000	0.000	0.000	**18.429**	18.890	0.000	0.000
Std. Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.301	**0.014**	0.000	0.000
Avg. Iteration	532.78	**252.96**	343.42	**183.98**	**354.10**	446.42	3000	3000	397.64	**177.08**
SR	50/50	50/50	50/50	50/50	50/50	50/50	0/50	0/50	50/50	50/50

TABLE XV(C): RESULTS FOR IOPSO AND MPSO FOR THE BENCHMARK WITH DIMENSION OF 30

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Method	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*	*mPSO*	*IOPSO*
Mean Fitness	0.000	0.000	0.000	0.000	0.000	0.000	**28.586**	28.881	0.000	0.000
Std. Dev.	0.000	0.000	0.000	0.000	0.000	0.000	0.273	**0.020**	0.000	0.000
Avg. Iteration	562.60	**289.86**	370.06	**219.28**	**395.22**	655.28	3000	3000	415.02	**206.88**
SR	50/50	50/50	50/50	50/50	50/50	50/50	0/50	0/50	50/50	50/50

TABLE XV(D): RESULTS FOR IOPSO AND MPSO FOR SCHAFFER'S F6

Problem	Schaffer's f6	
Method	*mPSO*	*IOPSO*
Mean Fitness	0.00e+00	0.00e+00
Std. Dev.	0.00e+00	0.00e+00
Avg. Iteration	**67.98**	97.74
SR	50/50	50/50

VI. Conclusion

The original PSO (OPSO) introduced in 1995 has been improved upon in this paper and has been named improved OPSO (IOPSO), without additional parameter(s) or complex computational efforts. Several experiments were performed using different nonlinear optimization problems well studied in literature with varied complexities and dimensions to compare the performance of IOPSO with the performances of six recent efficient PSO variants. From the experiments conducted, results show that IOPSO is very consistent in convergence velocity, convergence accuracy, global search ability and robustness than all the OPSO variants adopted for comparisons. IOPSO works well with swarm size not greater than 40. However, with high number of iterations it can comfortably work with higher swarm sizes.

Allowing the velocity limits and solution search space of particles to vary dynamically relative to the values of particles' dimension has greatly improved the performance of OPSO algorithm. This was as a result of better exploration and exploitation activities of the algorithm with flexibility in concentrating on the promising areas in the solution search space for further search by the particles instead of the entire space all the time.

Further study is needed on the optimization of Rosenbrock because the algorithm still experienced premature convergence solving the problem. Also, further empirical investigation of the effect of noise on the performance of the proposed algorithm by using more optimization problems with noise is needed. Furthermore, a scalability study will be conducted by using the algorithm on problems with dimensions greater than 100. Finally, applying the proposed algorithm to real-world problems will be investigated.

Appendix 1

Experimental results obtained for Original PSO algorithm when velocity limits and search space limits were kept static.

Iteration 0: Initialization state

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	-24.9973	-2.6798	-27.3189	31.2818	1.2440	-23.7598	-20.7054	29.3716	15.8120	-31.7267	21.5945
2	9.7768	-6.6610	2.1409	15.4727	-9.5688	6.6259	-15.1245	-20.8422	-18.0165	-20.4337	20.5614
3	3.4141	13.1284	12.5347	-31.6679	11.1312	5.0291	-4.5519	-8.8463	6.3953	18.6167	20.6850
4	23.3647	-31.8631	15.3881	4.2457	-18.6393	30.3913	-1.9281	-26.3176	-4.5832	26.1878	21.6350
5	-20.8222	25.7218	4.7624	-4.6332	15.9622	24.5665	3.9390	4.7236	-30.3351	-4.2180	21.0533
6	-25.0429	-10.2343	27.7369	12.2075	-15.9793	-10.2171	-27.5820	-26.3737	8.2654	12.3757	21.2902
7	-5.5478	25.3483	-27.9825	31.6348	22.2695	25.2856	5.7528	19.7386	-31.2892	29.1800	21.7068
8	20.1485	-1.6873	28.4581	-10.7107	-5.0774	10.5479	-7.5564	8.6199	-13.2508	-7.5515	20.6966
9	16.0300	-15.3206	-27.0845	11.0854	13.4676	-3.6559	10.5331	25.2651	-18.9965	26.0282	21.0384
10	0.8420	29.1329	0.9894	8.2623	21.2772	-10.8109	-15.4903	-15.3400	-26.6860	-3.1638	20.8751

Iteration 10

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	-9.5775	8.5050	13.6811	-3.6087	17.6482	-8.0625	2.3476	22.1802	-2.4674	-26.7557	20.8155
2	-10.7869	7.4366	14.2270	17.9172	-7.6638	-6.6673	31.9691	-23.8333	-0.6684	-7.8016	20.7677
3	20.6746	-3.5014	-17.6140	-18.6915	-31.8585	2.8360	-14.9008	-15.1372	19.8502	8.0729	20.9145
4	24.0232	-3.1183	8.8484	29.2548	7.6398	21.4961	7.4836	-25.6657	-16.2618	-29.9331	21.3401
5	6.4589	-29.4998	3.6289	-19.1809	13.7074	0.2264	24.1318	-15.7341	-22.4759	3.1417	21.2194
6	31.8361	-19.3649	-21.9912	8.7731	17.3316	24.6280	1.9834	-5.7799	-25.4585	-21.6496	21.3981
7	-15.3417	-6.5091	-28.6087	5.3995	-28.4838	17.7482	-7.5352	-9.6965	-23.8764	-18.8277	21.5336
8	-6.9055	17.1036	23.1164	21.3556	-22.0255	24.0964	-14.5442	23.3977	-25.9140	8.5283	21.1665
9	-5.4536	-24.6376	20.2403	20.1042	7.7701	24.5649	-7.8313	-10.7024	-28.3579	10.1072	21.2730
10	-0.3972	-20.9702	-19.4958	14.0257	7.4597	-0.9172	-6.9416	-4.0414	-28.0087	8.0745	20.0563

Iteration 20

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	3.5731	-5.7818	-19.2758	11.7207	19.4027	-17.6967	-9.1958	24.2263	-22.2842	-19.0468	21.1082
2	2.2476	-19.4430	-26.7984	-0.5095	4.0055	18.0942	29.6159	-28.3571	-31.8930	-8.1363	21.3553
3	30.3007	9.3716	18.7880	18.2364	-4.3004	-23.2864	14.8461	22.9884	-9.6773	1.5268	21.2377
4	1.2976	9.7296	-2.3934	7.2303	0.3418	-31.8191	3.1731	-0.1895	8.1259	18.3452	20.1208
5	2.4065	27.8393	-3.1227	4.0533	-19.6425	-1.0798	-11.7057	8.5010	16.8957	13.5931	20.3772
6	13.1445	-1.6784	-29.4500	-5.0847	-12.2327	30.7924	8.0393	7.6105	-3.0842	-1.7470	20.5931
7	-11.1974	-5.0874	-30.7542	18.4400	16.8934	-31.0622	12.5464	6.0303	-18.6541	-12.9351	20.9408
8	-19.1001	1.0887	14.5515	29.1161	1.6738	-19.2266	5.8677	23.6749	-0.9904	21.2992	20.8181
9	22.8798	-23.2605	-11.3066	31.7077	10.5584	19.6958	16.1144	-7.1917	15.6423	17.2987	21.3521
10	-1.9144	8.3954	-4.0554	28.5616	23.4996	9.0947	7.9905	-9.8366	-7.9165	18.0803	20.2351

Iteration 50

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	6.4118	26.9190	0.9788	-2.6566	13.5709	-26.3595	5.4654	-15.5886	15.0924	-17.2695	21.0460
2	-9.8258	30.4402	-26.1851	4.7184	-30.1112	8.4986	26.1265	-9.4668	21.8231	-1.6354	21.4495
3	-0.4352	5.1495	-20.9570	-13.9545	19.9318	13.7724	-11.8027	-13.9019	14.3579	-12.4680	20.2327
4	-16.7978	5.4568	-8.4531	7.6304	-28.6542	-30.3529	9.5790	-3.4640	2.2265	-12.6519	21.2794
5	-8.4113	10.7908	14.9354	13.8923	22.5234	7.3198	24.0203	8.4986	2.9928	23.8971	20.6445
6	-3.8409	-25.4197	-1.4372	6.8332	-18.1763	-11.4118	14.4304	7.4268	21.6938	1.2369	20.7324
7	-2.3379	27.0719	-21.6785	-3.8542	-0.8993	-11.8634	8.6957	7.7003	-11.1161	10.2275	20.0173
8	-1.1889	1.7352	-25.9687	10.5012	-0.4692	14.0572	16.7618	-5.9988	2.6411	-16.4312	20.1220
9	9.1273	2.8833	-0.4603	10.7812	16.7900	7.3565	20.9360	-11.5087	21.5833	-30.7529	20.9310
10	-0.0156	0.6126	-16.0604	25.9209	20.6494	-30.9606	-24.2338	-9.5977	1.5047	3.9225	20.9205

Iteration 100

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	11.3871	-10.7803	-27.6744	7.0541	-20.3879	-14.1282	30.4589	-21.8337	-3.5679	9.7338	21.3008
2	-5.5953	-6.1570	-17.0900	14.3686	6.8755	29.1574	-10.9838	16.2174	-0.4425	-1.3969	20.3540
3	-4.1888	-6.4975	-16.1131	23.0767	-20.4793	14.5525	-26.5726	-2.9188	12.7322	29.5546	21.3478
4	0.5645	11.5082	-0.1308	2.3655	11.7916	18.8184	-20.5615	29.4515	20.0930	12.6901	21.0908
5	23.3968	-1.3666	-24.4680	1.6752	-1.3551	0.0097	21.4358	-2.9025	20.9001	-24.1938	21.0875
6	30.3577	25.2763	10.0381	-24.3547	14.0642	-9.9417	-18.5338	0.0519	31.4084	-11.9425	21.2186
7	18.1247	-20.8702	-19.6981	-20.3142	7.4203	-13.3663	31.1404	-13.5926	7.0499	7.0947	21.0462
8	0.8815	0.0132	17.3364	-6.4547	31.9944	31.3149	-13.5031	-22.8926	4.4094	-8.1189	21.0956
9	-3.4764	-29.2753	-15.5202	15.8452	6.1010	22.5230	10.7976	-24.5199	7.2464	15.4126	21.3428
10	-2.7331	-28.9624	-18.7217	0.5734	15.5571	-1.2504	-7.1333	23.3516	19.3271	-24.8294	21.1995

Appendix 2

Experimental results obtained for Original PSO algorithm when velocity limits and search space limits were allowed to vary dynamically.

Iteration 0: Initialization state

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	-3.5250	0.2855	-28.7509	-20.2534	17.1054	31.3762	6.7659	-14.7401	9.0177	-23.3386	21.2897
2	21.0522	30.3839	5.4546	1.7135	-13.9940	20.0352	-28.2359	-23.4844	26.8322	-15.6490	21.4117
3	-5.5858	17.4756	31.2305	19.6098	-22.7965	-24.1239	-14.9091	-31.7354	-25.8181	0.5503	21.5651
4	27.6122	-1.7497	-9.5722	-7.3333	-16.6750	23.8907	-8.8003	-5.0156	12.7990	21.3582	20.9327
5	2.9442	-28.5006	-29.6663	-5.7181	-29.3814	-31.8090	-31.4788	14.2763	13.8185	-21.8334	21.6477
6	16.6764	22.1066	-12.5344	-15.1763	-27.9602	-28.4455	-0.5211	24.5660	30.5120	17.7648	21.6973
7	23.3694	8.9234	13.1659	2.6325	-21.8978	-18.5289	9.9891	-14.2069	-14.7211	29.9331	20.9023
8	0.7121	-24.0447	15.7643	28.0537	11.2289	2.2699	-25.9862	16.5780	13.7474	-6.1322	20.7731
9	-6.4346	-18.3142	7.2870	-31.4211	-22.7639	-25.6519	27.4261	-30.1715	-9.8428	13.2883	21.7027
10	-24.5427	25.4112	4.0457	23.4111	-2.3141	-22.3732	-23.9261	24.4092	13.2537	-14.2995	21.6029

Iteration 10

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	11.0394	15.9085	-17.5174	4.7708	1.3686	-6.3809	9.0679	6.7526	15.5767	-5.6932	19.4455
2	-7.4505	-12.2549	-8.9327	-11.6891	-13.1298	-17.5964	-7.3384	2.2458	-0.4147	-10.6630	19.4263
3	3.6525	-3.2172	-7.4511	-11.7514	-5.5937	-4.8882	-15.8480	6.8306	-16.6976	0.7473	18.6050
4	2.6253	-5.0526	-11.3615	-5.1466	9.1389	-6.6932	5.7634	-9.0284	-13.5493	-4.3420	17.6431
5	-16.6376	-15.4981	-5.0560	10.4171	-10.3647	7.7989	13.7402	-4.6192	-14.9488	-14.0683	19.9987
6	-10.2750	-13.9208	2.6315	-2.8811	7.9265	-3.4148	6.4817	11.9053	-6.3526	-9.5545	18.0700
7	-8.2042	-15.4341	5.7788	-2.8420	7.4927	13.5079	16.7780	13.8853	-4.7326	11.2348	19.5904
8	-5.6884	-15.9813	-3.7365	-11.8840	-13.5568	1.4792	-13.9724	1.6633	16.2847	2.4187	19.3760
9	-0.9571	6.4425	4.1080	15.0421	-4.7890	0.7348	9.8927	9.1909	2.7381	-3.2575	16.5807
10	3.4297	11.5294	14.9956	13.6765	-13.8091	16.4678	-3.6056	15.4574	-16.6072	-6.2012	20.4427

Iteration 20

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	2.6215	0.8121	-5.3739	0.8147	4.0689	1.8807	-4.6498	3.7051	5.1376	-1.2619	11.6569
2	7.3537	6.7580	-2.7140	5.7055	-1.4706	-6.0146	0.5742	-7.7269	-9.1337	7.0022	15.8365
3	-0.5468	-8.3482	-6.4349	2.3879	-2.6680	0.5856	6.5631	-3.8257	-9.1224	-8.0598	15.6727
4	3.7529	-3.2566	-8.7767	2.0113	1.0031	-5.6818	3.9923	5.0050	3.8234	0.0796	12.7790
5	7.4133	0.6540	9.0956	1.0387	3.8049	3.1753	2.8417	5.1514	-8.2874	-7.6847	15.1583
6	-8.2273	-4.0256	-3.4603	3.8939	-8.1725	1.7190	-2.0037	0.5151	-4.7942	-5.8713	14.0053
7	4.5683	8.4266	3.5768	3.9468	4.2359	2.3334	6.2840	4.5253	-3.5746	-5.0394	14.5192
8	-1.6614	4.8199	-2.2997	0.0373	8.8395	7.0762	-8.1932	0.8033	7.8337	3.4074	14.7695
9	7.9683	-7.8389	-5.7627	7.8130	-9.0832	-0.4985	-9.0470	-3.4200	1.9069	7.5611	16.3330
10	5.5725	-6.0913	0.5485	0.4440	4.7620	8.2374	6.3278	-4.1018	6.0534	-6.8055	15.0380

Iteration 50

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	-0.4716	0.8854	-0.5727	0.2673	1.2188	0.9980	-0.6318	-0.6788	-0.0120	1.3228	4.7542
2	0.7376	-1.1127	0.9071	0.0546	0.5731	-0.0926	0.4955	-1.0368	-0.9398	0.0829	3.8764
3	-0.3496	-0.4373	-1.0073	1.0448	0.8889	-0.7114	-0.4469	-0.0849	0.0701	-0.0806	3.7700
4	-0.5756	-0.1973	-0.1454	0.5611	0.3107	-0.5367	-0.4946	-1.3852	0.2511	-0.3141	4.2794
5	-0.8205	-0.2171	1.0177	0.8656	-1.3132	-0.7781	-0.2522	-0.2427	0.3600	0.2550	4.2376
6	-1.0117	-0.2055	0.3291	0.0465	-0.4973	-1.2741	0.5070	-0.2792	0.7641	0.6904	4.2964
7	-0.0286	0.6929	0.6368	1.3085	0.5128	-0.3048	0.2724	-1.0397	1.1346	-0.1300	4.4083
8	-1.1295	0.4951	-0.0481	0.9059	-0.4823	1.2532	-0.2152	-0.4872	0.9723	0.2814	4.4175
9	0.7806	1.3273	1.0171	1.0549	-0.3087	1.0156	0.7134	1.3396	-0.3273	0.2928	4.9379
10	0.1961	1.2688	-0.7636	1.0923	1.0123	-0.1278	0.3859	1.0949	-1.0515	0.3774	4.4453

Iteration 100

Particle	Dimension (number of variables)										Fitness
	1	2	3	4	5	6	7	8	9	10	
1	-0.0016	-0.0343	0.0105	-0.0397	-0.0563	-0.0066	-0.0603	-0.0543	-0.0506	-0.0283	0.2434
2	0.0363	0.0001	-0.0202	0.0432	0.0457	0.0403	0.0032	0.0463	-0.0307	-0.0311	0.1944
3	-0.0381	0.0449	0.0420	-0.0544	-0.0487	-0.0460	0.0488	-0.0487	0.0012	0.0128	0.2592
4	-0.0119	0.0390	-0.0161	-0.0515	0.0206	0.0546	-0.0341	-0.0467	0.0320	0.0287	0.2140
5	-0.0293	0.0052	0.0183	-0.0533	-0.0459	0.0471	-0.0190	0.0445	0.0131	0.0537	0.2205
6	0.0013	0.0416	0.0168	0.0600	0.0445	0.0112	-0.0118	0.0072	-0.0555	0.0607	0.2297
7	0.0091	0.0529	0.0022	-0.0057	0.0549	0.0617	-0.0432	-0.0333	-0.0419	0.0556	0.2589
8	0.0167	-0.0078	-0.0570	-0.0017	-0.0595	0.0212	-0.0379	0.0308	0.0560	-0.0312	0.2236
9	0.0375	-0.0602	0.0028	0.0120	-0.0530	0.0043	-0.0404	-0.0099	-0.0315	-0.0551	0.2202
10	0.0087	0.0232	-0.0557	0.0358	-0.0449	-0.0176	0.0607	-0.0405	-0.0437	-0.0431	0.2476

REFERENCES

[1] Akbari, R. and Ziarati, K. (2011). A rank based particle swarm optimization algorithm with dynamic adaptation. Journal of Computational and Applied Mathematics, Elsevier, 235, 2694–2714,

[2] Arasomwan, A. M. and Adewumi, A. O. (2013), On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization, The Science World Journal, In press.

[3] Eberhart, R. C. and Kennedy, J., (1995). A new optimizer using particle swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95. (Nagoya, Japan, 1995), pp. 39-43.

[4] Eberhart, R. C. and Shi, Y. (2002). Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 Congress on Evolutionary Computation, Korea, 1, pp. 94–100.

[5] Evers, G.I. (2009). An automatic regrouping mechanism to deal with stagnation in particle swarm optimization", MSc. Thesis, Graduate school of the University of Texas-Pan American, May, 2009

[6] Gao Y.-l. and Duan, Y.-h., (2007), A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy, In Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on, pp. 199-203.

[7] Yang G., Zhang, D. and Hu X. (2010). A Learning Algorithm Based on PSO and L-M for Parity Problem, In, Chris Myers (Ed.) Stochastic Control, InTech, pp. 151-166. Retrieved from: http://www.intechopen.com/books/stochastic-control

[8] Guochao, N., Baodi, C. and Jianchao, Z. (2010). Repulsive Particle Swarm Optimization based on new diversity. Control and Decision Conference (CCDC), pp. 815-819.

[9] Jianbin Xin, Guimin Chen, Yubao Hai (2009). A Particle Swarm Optimizer with Multi-Stage Linearly-Decreasing Inertia Weight. Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO 2009), pp 505-508.

[10] Jing, B., Xueying, Z. and Yueling, G. (2009). Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system. Proceedings of the International Conference on Mechatronics and Automation (ICMA 2009), pp. 4754-4759.

[11] Karaboga, D. and Akay B. (2009). A Comparative Study of Artificial Bee Colony Algorithm. Applied Mathematics and Computation, 214 (1), pp. 108-132.

[12] Kennedy, J. and Eberhart, R.C. (1995). Particle swarm optimization. In Proceedings of IEEE international conference on neural networks, 4, Perth, Australia. pp. 1942–1948.

[13] Liu, B. , Wang, L. , Jin, Y. , Tang, F. and Huang, D. (2005). Improved particle swarm optimization combined with chaos, Chaos, Solution and Fractals, 25, pp. 1261-1271.

[14] Liu, H., Su, R., Gao Y., and Xu R., (2009), Improved Particle Swarm Optimization Using Two Novel Parallel Inertia Weights. IEEE Second International Conference on Intelligent Computation Technology and Automation, pp 185-188.

[15] Omran M.G.H. (2009). Using Opposition-based Learning with Particle Swarm Optimization and Barebones Differential Evolution. In Aleksandar Lazinica (Ed), Particle Swarm Optimization, InTech, pp. 373-384, Retrieved from: http://www.intechopen.com/books/particle_swarm_optimization

[16] Malik, R.F. Rahman, T.A. Hashim, S.Z.M. and Ngah, R. (2007). New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight, International Journal of Computer Science and Security, 1(2), pp. 35.

[17] Mansour, M.M., Mekhamer, S.F. and El-Kharbawe, N.E.S. (2007), A Modified Particle Swarm Optimizer for the Coordination of Directional Overcurrent Relays, IEEE Transactions on Power Delivery, Vol. 22, pp. 1400-1410.

[18] Martins A. A. and Oluyinka A. A. (2013), An Adaptive Velocity Particle Swarm Optimization for High-Dimensional Function optimization, IEEE Congress Evolutionary Computation (CEC), 2013, pp. 2352-2359

[19] Dieu, V.N, Schegner, P., and Ongsakul, W. (2011), A newly improved particle swarm optimization for economic dispatch with valve point loading effects, in IEEE Power and Energy Society General Meeting, pp. 1-8.

[20] Nickabadi A., Ebadzadeh M. M., and Safabakhsh R. (2011), A novel particle swarm optimization algorithm with adaptive inertia weight. Applied soft computing, 11, 3658-3670.

[21] Poli R., Kennedy J., Blackwell T. (2007). Particle swarm optimization: An overview. Swarm Intelligence. Vol. 1, pp 33–57.

[22] Quin, Z., Yu, F., Shi, Z. and Wang, Y. (2006), Adaptive inertia weight particle swarm optimization. L. Rutkowski et. al. (Eds), Proceedings of the 8th International Conference Artificial Intelligence and Soft Computing (ICAISC 2006), Zakopane, Poland, June 25-29, pp. 450-459

[23] Shi, Y. H., Eberhart, R. C., (1998), A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, pp. 69-73.

[24] Shi, Y. and Eberhart, R., (1998), Parameter selection in particle swarm optimization. Proceedings of the 7th International Conference on Evolutionary Programming (EP98) San Diego, California, USA, March 25–27, Vol. 1447, pp. 591-600.

[25] Shi, Y. H., and Eberhart, R. C. (2001), Fuzzy adaptive particle swarm optimization. Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001), Korea, Vol. 1, pp. 101 – 106.

On Adaptive Chaotic Inertia Weights in Particle Swarm Optimization

[1]Martins Akugbe ARASOMWAN and [2]Aderemi Oluyinka ADEWUMI
School of Mathematics, Statistics and Computer Science
University of Kwa-Zulu-Natal
Durban, South Africa
[1]accuratesteps@yahoo.com, [2]adewumia@ukzn.ac.za

***Abstract*—Inertia weight is one of the control parameters that influence the performance of Particle Swarm Optimization (PSO). Since the introduction of the inertia weight parameter into PSO technique, different inertia weight strategies have been proposed to enhance the performance of PSO in handling optimization problems. Each of these inertia weights has shown varying degree of efficiency in improving the PSO algorithm. Research is however still ongoing in this area. This paper proposes two adaptive chaotic inertia weight strategies based on swarm success rate. Experimental results show that these strategies further enhance the speed of convergence and the location of best near optimal solutions. The performance of the PSO algorithm using proposed inertia weights compared with PSO using the chaotic random and chaotic linear decreasing inertia weights as well as the inertia weight based on decreasing exponential function adopted for comparison in this paper are verified through empirical studies using some benchmark global optimization problems.**

***Keywords*—Adaptation; Success rate; Inertia weights; Chaotic; Swarm Intelligence; Global optimization; Particle swarm optimization**

I. INTRODUCTION

Since the inception of PSO strategy for solving optimization problems, a lot of work has been done by researchers to enhance its efficiency in handling optimization problems. The PSO has a small number of parameters that regulates the behaviour of the algorithm. These include particle swarm size, problem dimensionality, particle velocity, inertia weight, cognitive learning rate and social learning rate. The inertia weight parameter (popularly represented as ω) introduced in [3], is the most important compared with other parameters [4]. The motivation behind its introduction was the desire to better control (or balance) the scope of the (local and global) search and reduce the importance of (or eliminate) velocity clamping, $V_{\max}$ during the optimization process [2, 5, 8]. According to [8], the inertia weight was successful in addressing the first objective, but could not completely eliminate the need for velocity clamping. As reported in [5, 6], ω gets important effect on balancing the global search and the local search in PSO. Therefore, the feature of the divergence or convergence of particles can be controlled only by parameter ω, however, in conjunction with the selection of values for the acceleration constants [7, 8].

Each individual in the particle swarm is composed of three n-dimension vectors (current position, previous position, and velocity), where n is the dimensionality of the search space. Thus, in a physical n-dimensional search space, the position and velocity of each particle i are represented as the vectors $X_i = (x_{i1}, \ldots, x_{in})$ and $V_i = (v_{i1}, \ldots, v_{in})$, respectively. In the course of movement within the search space in search of the optimum solution of a problem, the particle's *velocity* and *position* are updated as shown in equations (1) and (2)

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (Pbest_i^k - X_i^k) + c_2 r_2 (Gbest_i^k - X_i^k) \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

where, c_1 and c_2 are acceleration (weighting) factors known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of *Pbest* (previous best) and *Gbest* (global previous best); r_1 and r_2 are random numbers between 0 and 1; k is the iteration index and ω is the inertia weight.

From the time PSO was proposed, several strategies have been proposed to modify the algorithm by using dynamic value of ω in each iteration [1, 4, 10, 11, 15, 21]. These strategies include random [9], chaotic random [14], linear decreasing [8], and chaotic linear decreasing [14] strategies which were adopted for comparison in this work. In [8], the linear decreasing inertia weight strategy decreases from a value of 0.9 to 0.4 in course of searching for solution to the problem being solved. Though it enhances the performance of PSO, it usually get into local optimum when solving functions with more apices [14]. In [9] it was experimentally found that random inertia weight strategy increases the convergence in PSO and could find good results with most functions. A chaotic term was included to the random as well as the linear decreasing inertia weight strategies in [14]. These strategies were experimentally proved to be superior to the random and linear decreasing strategies in terms of convergence speed, global search ability and convergence precision. The remaining part of the paper is organized as follows. In section 2, we give a summarized review of inertia weight strategies in PSO with more emphasis on the adaptive and chaotic inertia weights. Section 3 describes the proposed adaptive chaotic inertia weights while in section 4, the experimental settings are stated. The experimental results are given and discussed in section 5 and section 6 concludes the paper.

978-1-4673-6004-3/13/$31.00

II. INERTIA WEIGHT STRATEGIES FOR PSO

The basic PSO presented by [17] has no inertia weight. The first inertia weight was introduced into PSO in [3], this inertia weight was static in nature. However a lot of improvement on inertia weight strategies have been made over the years [3, 6, 8, 9, 11, 12, 13]. By reason of its operation, the inertia weight (ω) can be interpreted as the fluidity of the medium in which a particle moves [2]; showing that setting it to a relatively high initial value (for example, 0.9) makes particles move in a low viscosity medium and performs extensive exploration. Gradually reducing it to a much lower value (for example, 0.4) makes the particle moves in a high viscosity medium and performs more exploitation. Different strategies have been proposed to determineω. These strategies could be categorized into static and dynamic (or variable). In the static strategy, a fixed or constant value is used for the entire search duration [3]. The dynamic category is subdivided into random adjustment, linear time-varying, nonlinear time-varying and adaptive[1, 8]. In the random adjustment techniques, different inertia weight is randomly selected in each iteration [9, 21]. The linear time-varying can be subdivided into linear time decreasing and linear time increasing. In linear time decreasing approach an initially large inertia weight (commonly 0.9) is linearly decreased to a small value (commonly 0.4) [6, 11, 12, 13, 14]. There are cases where values other than 0.9 or 0.4 are used [1, 15, 16]. Linear time increasing deals with the reverse [19]. Also, nonlinear time-varying can be subdivided into nonlinear time decreasing, and nonlinear time increasing. For nonlinear time decreasing, an initially large value decreases nonlinearly to a small value [12, 15]. It allows a shorter exploration time than the linear decreasing methods, with more time spent on refining solutions [8]. Nonlinear time decreasing methods seem more appropriate for smoother search spaces [8]. Again, the nonlinear time increasing deals with the reverse [11]. The adaptive inertia weight approaches can be subdivided into fuzzy adaptive and non-fuzzy adaptive. Fuzzy adaptive inertia weight is dynamically adjusted on the basis of fuzzy sets and rules in each iteration [1, 8, 10]. The non-fuzzy adaptive inertia weights are dynamically adjusted based on the state of the swarm in terms of fitness, particle rank, and distance to particle, global best positions, and particle success rate which are regarded as feedback parameters [1]

A. *Adaptive inertia weight in PSO based on swarm success rate*

Generally, adaptive inertia weight strategies monitor the search space and adapt the inertia weight value based on one or more feedback parameters [1, 18]. Our focus is on the inertia weights based on swarm success rate feedback parameter. The PSO variant (AIWPSO) proposed in [1] monitors the search situation and adapts the inertia weight value based on the swarm success rate parameter. It can properly adapt the value of the inertia weight in the static and dynamic environment using (3). The best particle was mutated by adding a Gaussian noise with zero mean standard deviation to one of its randomly chosen dimension and used to replace the worst particle at the end of each iteration to improve on the exploration of the method.

$$\omega_t = (\omega_{start} - \omega_{end})SR + \omega_{end} \tag{3}$$

where *SR* is as shown in (5) and the range of inertia weight [ω_{start}, ω_{end}] was selected to be [0, 1].

AIWPSO was found work better than its competitors due to its adaptive nature. For static functions, ω changes over time, starting with a large value in the first few number of iterations due to the high rate of successful movements (particle best updates) which facilitates exploration and later converges to oscillate round about 0.39 (a value suitable for sphere function) for exploitation. For the dynamic function described in the paper, AIWPSO was able to locate (track) new optimum position after the environment changes (peak movements). At each peak movement the particles are placed on a point far from the optimum which is identified using the number of successful movements of the particles which grows to a value of about 1 (high ω) which help the particles to accelerate up towards the new optimum point and after that ω rapidly reduces to a value around 0.4. A high percentage of success indicates that the particles have converged to a point that is far from the optimum point and the entire swarm is slowly moving towards the optimum while a low percentage of success shows that the particles are oscillating around the optimum without much improvement [1].

The success of particle *i* at iteration *t*, in a minimization problem, is defined in (4):

$$succ_t^i \begin{cases} 1 & f(Pbest_t^i) < f(Pbest_{t-1}^i) \\ 0 & f(Pbest_t^i) \geq f(Pbest_{t-1}^i) \end{cases} \tag{4}$$

Where $pbest_t^i$ is the current best position of particle *i* until iteration *t* and *f( )* is the function to be optimized. The success rate (*SR*) of the swarm is computed using (5):

$$SR_t = \sum_{i=1}^{n} succ_t^i / n \tag{5}$$

Where *n* is the swarm size and $SR_t \in [0,1]$ is the percentage of the particles with improvement in fitness in the last iteration. Where no particle succeeds to improve its fitness, *SR* is set to *0*; where all the particles succeed in improving their individual fitness, *SR* is set to *1*. Therefore *SR* reflects the state of the swarm and serves as a feedback parameter to the PSO algorithm to determine the inertia weight at each iteration.

B. *Chaotic inertia weight in PSO*

Chaos is a form of nonlinear dynamic system which has the characteristic of stochastic property, ergodicity, and sensitive to initial value [20]. Two chaotic inertia weights were proposed in [14], they are Chaotic descending inertia weight (CDIW) and Chaotic random inertia weight (CRIW) shown in (7) and (8) respectively. The aim then was to improve on the random and linear descending inertia weights using logistic mapping shown in equation (6), to avoid getting into local optimum in searching process by utilizing the merits of chaotic optimization.

$$z = \mu \times z \times (1 - z) \tag{6}$$

where $3.75 < \mu \leq 4$ and when $\mu = 4$ its chaotic result sprinkles the interval of [0,1].

$$\omega_t = (\omega_{start} - \omega_{end})\left((T_{\max} - t)/T_{\max}\right) + \omega_{end} \times z \tag{7}$$

$$\omega_t = 0.5 \times rand() + 0.5 \times z \tag{8}$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, *rand()* is a uniform random number in [0,1], t is the current iteration, T_{max} is the maximum iteration, $z = 4 \times z \times (1 - z)$ is a logistic mapping and z is a random number in the interval of (0,1). The results from [14] shows that the proposed methods make PSO have preferable convergence precision, quick convergence velocity, and better global search ability. This is because, at the initial stage, it has a wide rough search (exploration) which makes it to quickly get to a position near the global optimum and at the later stage, it has a detailed search (exploitation) which make it to exploit the neighbourhood of the discovered near optimal position till discovering the optimum result or end of maximum iteration. With the chaotic characteristic they are able to get away from the local optimum unlike Linear decreasing inertia weight which during exploitation, becomes reduced such that the particles have not enough velocity to escape from local optimum. Shown in Fig. 1 is how ω of CDIW and CRIW changes. Due to non-repetition of chaos the algorithm with the ω can carry out overall searches at higher speed and diversify the particles and improves the algorithm's performance in preventing premature convergence to local minima compared with ω without chaos characteristic.

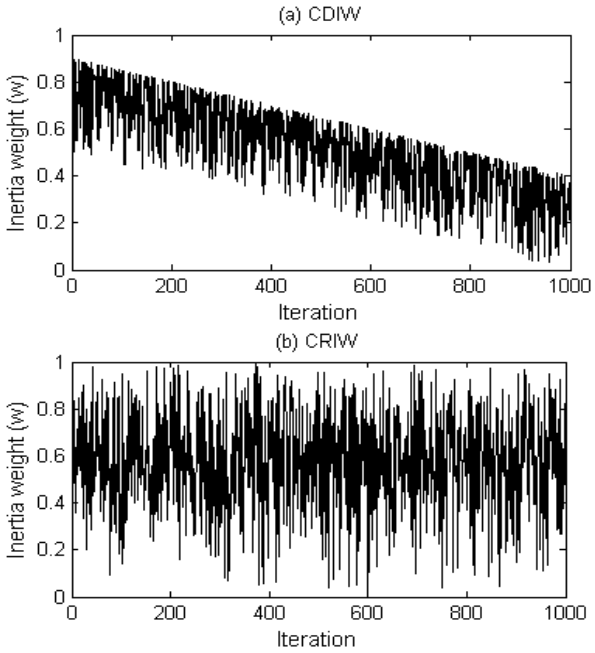


Fig. 1. Changes in inertia weights of CDIW and CRIW

C. Decreasing exponential function inertia weight in PSO

The method in [18] is based on decreasing exponential functions. It is made up of two parts (base and power). The PSO algorithm's iteration was used in these parts as shown in (9). As the iteration increases, ω goes from one to zero as shown in Fig. 2.

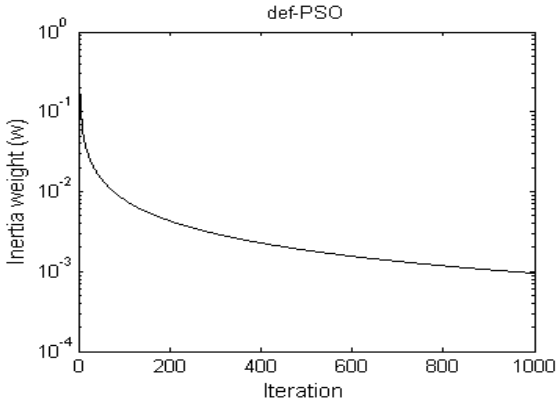


Fig. 2. Changes in inertia weight of decreasing exponential inertia weight

Graphical results in [18] show evidences of its superiority to the competitors in fitness quality and speed of convergence. Shown in Fig. 2 is the change of inertia weight over time.

$$\omega_t = t^{-t\sqrt{t}} \tag{9}$$

III. Proposed Adaptive Chaotic Inertia Weights

Our proposed adaptive chaotic inertia weights simply combine the swarm success rate feedback parameter with chaotic mapping to harness together adaptivity and chaotic characteristics. These inertia weights labelled CAIWS-D and CAIWS-R are shown in (10) and (11) respectively.

$$\omega_t = ((\omega_{start} - \omega_{end})\left((T_{\max} - t)/T_{\max}\right) + \omega_{end}) \times z \tag{10}$$

$$\omega_t = (0.5 \times SR + 0.5) \times z \tag{11}$$

where, $z = 4 \times SR \times (1 - SR)$. In this case z is not just a uniform random number in the interval of (0,1) but the swarm success rate in the interval [0,1]. Also in (11), *rand()* which is a uniform random number in the interval of [0,1] is replaced with *SR* in the interval [0,1]. ω_{start} and ω_{end} are the initial and final values of inertia weight respectively.

These proposed strategies are based on the fact that usually, a uniform random number does not convey much information about the state of the swarm and therefore cannot be assumed to adjust the inertia weight appropriately. However, the success rate of the swarm as a feedback parameter can help realize the state of the swarm in the search space and hence adjust the value of inertia weight in each iteration appropriately for better results. The values calculated by these strategies are always in the interval [0,1]. Giving below is the PSO pseudo-code of inertia weight adaptation using CAIWS-D and CAIWS-R.

Pseudo-code using CAIWS-D and CAIWS-R

Begin PSO Algorithm
Input: *f:* the function to optimize
ps: the swarm size
d: the problem dimension
Output: x^*: the best fitness value found
Initialize: $x_i = (x_{i1}, \ldots, x_{id})$ and $v_i = (v_{i1}, \ldots, v_{id})$, for all particles in problem space
evaluate $f(x_i)$ in d variables and get $pbest_i$,
$(i = 1, \ldots, ps)$
$gbest \leftarrow$ best of $pbest_i$
While stopping criteria is false **do**
succ $\leftarrow 0$
Begin Loop for *ps* times
Begin Loop for *d* times
calculate ω using (10) or (11)
update v_i for particle using (1)
check for velocity boundaries
End
update x_i for particle using (2)
validate for position boundaries
End
If $f(x_i) < f(pbest_i)$ then
$pbest_i \leftarrow x_i$
succ $\leftarrow$ succ + 1
end if
If $f(x_i) < f(gbest_i)$ then
$gbest_i \leftarrow x_i$
end if
compute swarm success rate using (5)
End while
$x^* \leftarrow gbest$
Return x^*
End PSO Algorithm

Fig. 3 (a) and (b) shows the change of ω in CAIWS-D and CAIWS-R over time. Their structures reflect the rate of success of the particles over time.

IV. EXPERIMENTAL SETTINGS

The experiments were carried out in two stages. In *stage* 1, the results of CAIWS-D and CAIWS-R were compared with existing results in [14] while in *stage* 2, the method in [18], CDIW and CRIW were implemented and their performances compared with that of CAIWS-D and CAIWS-R. All experiments were done on a laptop computer with a 2.0GHz Intel Pentium dual-core processor, 2.0GB of RAM, running Windows Vista Home Basic. The simulation program was development in Microsoft Visual C# programming language, 2008 Express Edition.

A. Settings for stage 1of experiments

The settings used for CDIW and CRIW in [14] were also used to implement CAIWS-D and CAIWS-R for the purpose of fairness in comparing their performances.

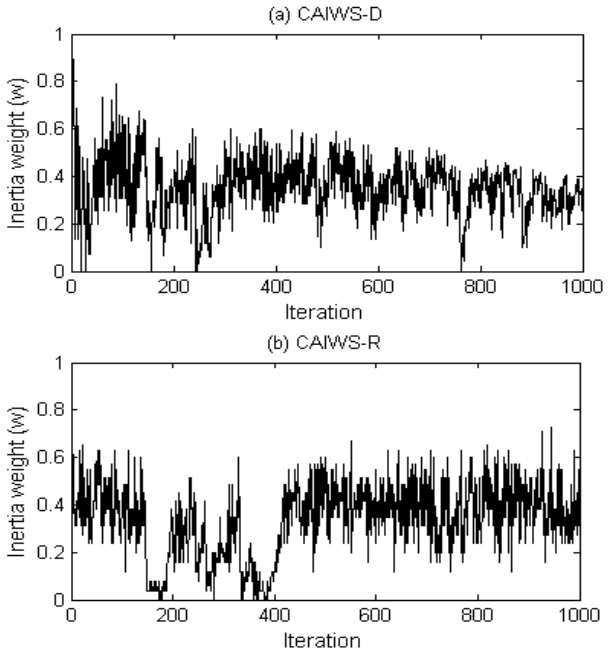


Fig. 3. Changes in inertia weights of CAIWS-D and CAIWS-R

B. Settings for stage 2 of experiments

CAIWS-D and CAIWS-R were implemented along side with the method in [18]. They were subjected to the settings and were used to optimize the same problems in [18]. A method is successful if at the end of a run its mean fitness value is not greater than 0.01 for Sphere and Griewank problems and 50 for Rastrigin problem.

Also, CAIWS-D and CAIWS-R were implemented together with the following four inertia weight strategies in [14].

a) $\omega_t = 0.5 \times rand() + 0.5$

b) $\omega_t = 0.5 \times rand() + 0.5 \times z$

c) $\omega_t = (\omega_{start} - \omega_{end})((T_{\max} - t)/T_{\max}) + \omega_{end}$

d) $\omega_t = (\omega_{start} - \omega_{end})((T_{\max} - t)/T_{\max}) + \omega_{end} \times z$

The following nine well-known benchmark problems, which are extensively used in the literature for the evaluation of metaheuristics were used.

Ackley (f_1): $f_1(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$

Search space: [-30,30], optimal value = 0

Griewank (f_2): $f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$

Search spae: [-600,600], optimal value = 0

Levy (f_3): $f_3(\vec{x}) = \left(\frac{\pi}{d}\right)\left(10\sin^2(\pi y_1) + \sum_{i=1}^{d-1}(y_i - 1)^2\left(1 + 10\sin^2(\pi y_{1+1})\right) + (y_d - 1)^2\right)$

Search space: [-10,10], optimal value = 0

Rastrigin (f_4): $f_4(\vec{x}) = \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$

Search space: [-5.12,5.12], optimal value = 0

Rosenbrock (f_5): $f_5(\vec{x}) = \sum_{i=1}^{d-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$

Search space: [-5,10], optimal value = 0

Schwefel (f_6): $f_6(\vec{x}) = \sum_{i=1}^{d} -x_i \sin\left(\sqrt{|x_i|}\right)$

Search space: [-500,500], optimal value = -418.9829d

Schwefel P2.22 (f_7): $f_7(\vec{x}) = \sum_{i=1}^{d} |x_i| + \prod_{i=1}^{d} |x_i|$

Search space: [-10,10], optimal value = 0

Sphere (f_8): $f_8(\vec{x}) = \sum_{i=1}^{d} x_i^2$

Search space: [-100,100], optimal value = 0

Step (f_9): $f_9(\vec{x}) = \sum_{i=1}^{d} \left(|x_i + 0.5|\right)^2$

Search space: [-10,10], optimal value = 0

The number of variables (dimensions) for all problems was set to 30 because it is commonly used in literature, for example in [1, 6, 13, 14, 16, 18]. In all experiments, the maximum number of iterations and swarm size were set to 2,000 and 50 particles, respectively. The stop condition is reaching the maximum number of iterations. The values of ω_{start} and ω_{end}, were set to 0.9 and 0.4, c_1 and c_2 were set to 2.0 as used in [14]. V_{max} was set relative to the testing problems using $V_{max} = \delta(X_{max} - X_{min})$, where X_{max} is the maximum value of the domain of X, X_{min} is the minimum value of the domain of X, $\delta \in (0,1]$ was set to 0.5 based on the findings of [19]. All experiments were repeated ten times. The performance of each method takes into account the average best solution and standard deviation of solution found in each run.

V. Results and Discussions

The results are presented in stages in line with stages with which the experiments were carried out. Presented in *stage 1* is the performance between CAIWS-D and CAIWS-R with the results of CRIW and CDIW recorded in literature. In *stage 2*, the results of def-PSO, CRIW, CDIW, CAIWS-D and CAIWS-R as implemented in this work are reported.

A. Results of stage 1 of experiments

Shown in Table I are the results of CAIWS-D and CAIWS-R with the results of CRIW and CDIW recorded in literature. From the results, the mean fitness indicates that CAIWS-D and CAIWS-R can get better optimum fitness value with preferable convergence precision and quick convergence speed in Schaffer f6 and Sphere problems than CRIW and CDIW. They also demonstrate better stability, robustness and global search abilities in the two problems as indicated by the standard deviation and success rate. But they lost superiority in Griewank, Rastrigin and Rosenbrock problems to their competitors, in every way. The reason for this is that CAIWS-D and CAIWS-R were not able to perform thorough global search during some of the runs which affected their overall performances.

TABLE I. The Mean, Standard Deviation and Success Rate of CDIW, CRIW, CAIWS-D and CAIWS-R over 500 Independent Runs

Function	Performance index	Inertia Weight PSO			
		Linear Decreasing		*Random*	
		CDIW	*CAIWS-D*	*CRIW*	*CAIWS-R*
Griewank	Mean Fitness	0.014773	0.020989	0.016616	0.024682
	Standard Deviation	0.002955	0.073804	0.003323	0.075676
	Success Rate	96.2	93.4	98.2	90.8
Rastrigin	Mean Fitness	40.044561	61.862829	40.267957	63.740040
	Standard Deviation	8.028912	23.993071	8.053591	23.194192
	Success Rate	83.6	37.6	91.8	30.8
Rosenbrock	Mean Fitness	44.305058	5679.561129	37.090110	6382.479413
	Standard Deviation	8.861012	21321.202152	11.618022	22607.241308
	Success Rate	99.6	70.2	99.4	68.4
Schaffer f6	Mean Fitness	0.007732	0.004167	0.009211	0.004609
	Standard Deviation	0.001546	0.004803	0.001842	0.004848
	Success Rate	22	56.6	24.4	51.6
Sphere	Mean Fitness	0.000092	0.000000	0.000087	0.000000
	Standard Deviation	0.000016	0.000000	0.000017	0.000000
	Success Rate	100	100	100	100

B. Results of stage 2 of experiments

The performances of the method in [18], CAIWS-D and CAIWS-R are presented in Table II. Their comparisons are based on mean fitness, standard deviation, average iteration to reach goal and success rate.

TABLE II. The Mean, Standard Deviation and Success Rate of Inertia Weight in [18], CAIWS-D and CAIWS-R over 10 Independent Runs

Function	Performance index	Inertia Weight PSO		
		def-PSO	*CAIWS-D*	*CAIWS-R*
Griewank	Mean Fitness	0.103337	0.007341	0.017089
	Standard Deviation	0.134860	0.022023	0.034599
	Least Iteration	129	268	219
	Worst Iteration	280	305	357
	Average Iteration	187	290	248
	Success Rate	60	90	80
Rastrigin	Mean Fitness	65.968908	62.484002	52.520341
	Standard Deviation	13.529611	17.666654	16.395482
	Least Iteration	388	221	122
	Worst Iteration	388	390	368
	Average Iteration	388	306	203
	Success Rate	10	20	40
Sphere	Mean Fitness	0.000000	0.000000	0.000000
	Standard Deviation	0.000000	0.000000	0.000000
	Least Iteration	220	408	340
	Worst Iteration	462	465	380
	Average Iteration	342	434	359
	Success Rate	100	100	100

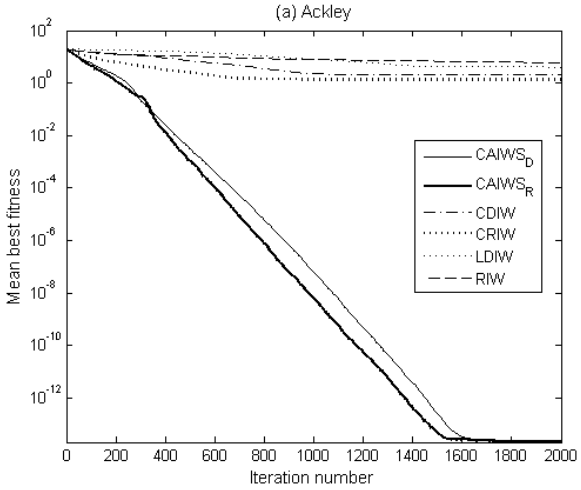
From the results in Table II, in Griewank and Rastrigin problems, CAIWS-D and CAIWS-R achieved better accuracy in mean fitness and demonstrate higher global search ability than def-PSO, the method in [18], despite the early

convergence in Griewank and better stability in Rastrigin demonstrated by def-PSO. For Sphere problem, the three methods had the same performance except that the average convergence speed of def-PSO is higher. Generally, the proposed method performed better than their competitor in multimodal problems.

Table III shows the results of applying the PSO algorithm which implements RIW, LDIW, CRIW and CDIW with the proposed CAIWS-D and CAIWS-R. The results are compared based on the final accuracy of the averaged best solutions as well as standard deviation of the best solutions. The mean best fitness (Mean) is a measure of the precision that the algorithm can get within given iterations while the standard deviation (SD) is a measure of the algorithm's stability and robustness.

CAIWS-R is compared with RIW and CRIW while CAIW-D is compared with LDIW and CDIW and then all the strategies are finally compared together. The bold values indicate the best fitness solution. The results show that CAIWS-R provides the best accuracy, compared with RIW and CRIW, in all the test problems considered except in f_9 where CAIWS-R and CRIW are at the same level of accuracy, stability and robustness. Also CAIWS-D provides the best accuracy, compared with LDIW and CDIW, in all the test problems considered except f_2 where LDIW performs better in fitness accuracy, robustness and stability. Comparing all the strategies together, CAIWS-R performed best followed by CAIWS-D. What gave the proposed inertia weight strategies edge over the competitors is the fact that the success rate of the swarm serves as a feedback parameter to help provide information on the state of the swarm in the search space which helps adjust the value of the inertia weight in each iteration appropriately for better results.

The convergence curves in Fig. 4 provide insight into the searching behaviour of the six strategies in Ackley and Rastrigin test problems. All methods in Fig. 4(a) have long periods of iterations in which the fitness is not improved much, except for CAIWS-D and CAIWS-R, which are more or less straight lines in logarithm scale for a long time. This shows that our proposed adaptive chaotic methods are consistent in their speed of convergence to the optimum and have outstanding performance over other strategies.



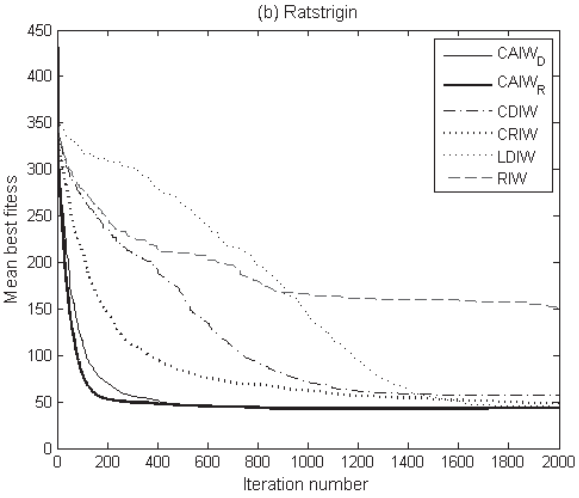


Fig. 4. The mean of the best fitness of 10 independent runs on Ackley and Rastrigin test problems

For Rastrigin problem in Fig. 4(b), the proposed methods also performed better than their competitors both in speed of convergence to the optimum and solution accuracy. What gave CAIWS-D and CAIWS-R superiority over their competitors is their adaptive nature.

TABLE III. THE MEAN AND STANDARD DEVIATION (SD) OF THE BEST FITNESS OF SIX INERTIA WEIGHT STRATEGIES ON NINE TEST PROBLEMS

Function		Random			Linear Decreasing		
		RIW	***CRIW***	***CAIWS-R***	***LDIW***	***CDIW***	***CAIWS-D***
f_1	Mean	6.0629E+00	1.3507E+00	**2.0900E-14**	4.0834E+00	1.9947E+00	**2.2200E-14**
	SD	5.7679E-01	4.2713E+00	5.2589E-15	6.5753E+00	6.3076E+00	5.4934E-15
f_2	Mean	4.3447E+00	1.9923E-02	**1.8149E-02**	**1.2025E-02**	9.0651E+00	2.0629E-02
	SD	8.5579E-01	2.1296E-02	2.1198E-02	1.6834E-02	2.8605E+01	2.0306E-02
f_3	Mean	4.7117E-01	1.2050E-01	**1.0724E-02**	2.4645E-01	3.9394E-01	**7.6291E-02**
	SD	5.0303E-01	2.5530E-01	3.3913E-02	3.2763E-01	3.3417E-01	2.0634E-01
f_4	Mean	1.5158E+02	4.8236E+01	**4.2587E+01**	4.4912E+01	5.7018E+01	**4.2966E+01**
	SD	4.5464E+01	2.4039E+01	1.3163E+01	2.3827E+01	2.3984E+01	1.2981E+01
f_5	Mean	5.6345E+04	5.6018E+03	**2.9732E+02**	7.2712E+04	2.7250E+04	**5.7041E+01**
	SD	7.2362E+04	1.7546E+04	7.7901E+02	6.9162E+04	3.8866E+04	9.5159E+01
f_6	Mean	-9.4355E+3	-9.5267E+3	**-9.5888E+3**	-8.6574E+3	-9.3759E+3	**-9.9136E+3**
	SD	8.7794E+02	8.9974E+02	6.5649E+02	5.1187E+02	8.5024E+02	3.1901E+02
f_7	Mean	8.8935E+00	1.0000E+00	**0.0**	1.2000E+01	6.0000E+00	**2.0000E+00**
	SD	5.7282E+00	3.1623E+00	0.0	9.1894E+00	8.4327E+00	6.3246E+00
f_8	Mean	3.7163E+02	1.1629E-08	**0.0**	5.4141E-12	1.0000E+03	**0.0**
	SD	9.5088E+01	1.3570E-08	0.0	4.5455E-12	3.1623E+03	0.0
f_9	Mean	6.4000E+00	**0.0**	**0.0**	0.0	0.0	0.0
	SD	2.7162E+00	0.0	0.0	0.0	0.0	0.0

C. Searching behaviour of CAIWS-D

In Fig. 5, the changes in success rate (SR), inertia weight (IW) of Griewank problem of two dimensions using 20 particles and their relationship are shown. At the beginning, both SR and IW were high, however, as SR drops below 0.4, the IW was sustained to fluctuate between 0.7 and 0.2 appropriate for the problem with the help of the chaos characteristics of the logistic mapping.

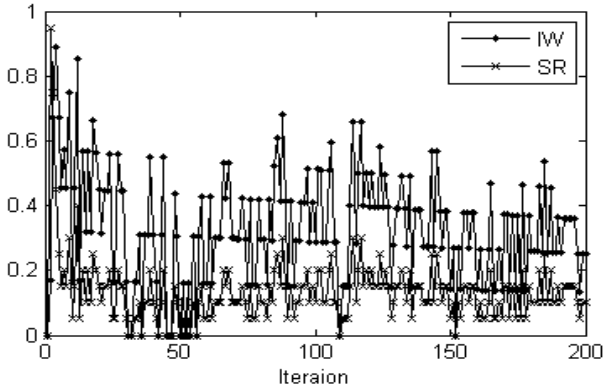
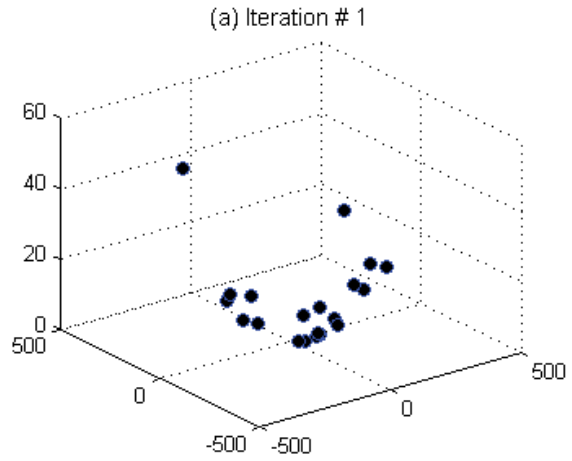


Fig. 5. Changes in success rate (SR) and inertia weight (IW) of CAIWS-D

For the purpose of convenient observation, the searching behaviour of the proposed algorithm with CAIWS-D for a group of 20 particles on a 2-dimensional Griewank problem is presented in Fig. 6(a) – (d). The figure represents the distribution of particles with different roles at the 1st, 50th, 100th, and 150th iterations. The initial positions of the particles were chosen at random before the iterations began. During the initial iterations, relative to the feedback from the success rate of particles in the swarm, the particles are in the state of exploration as shown, for example, in Fig. 6(a). As the iterations increases, the algorithm begins to exploit around the near optimal solution discovered during the exploration stage as shown, for example, in Fig. 6(d).



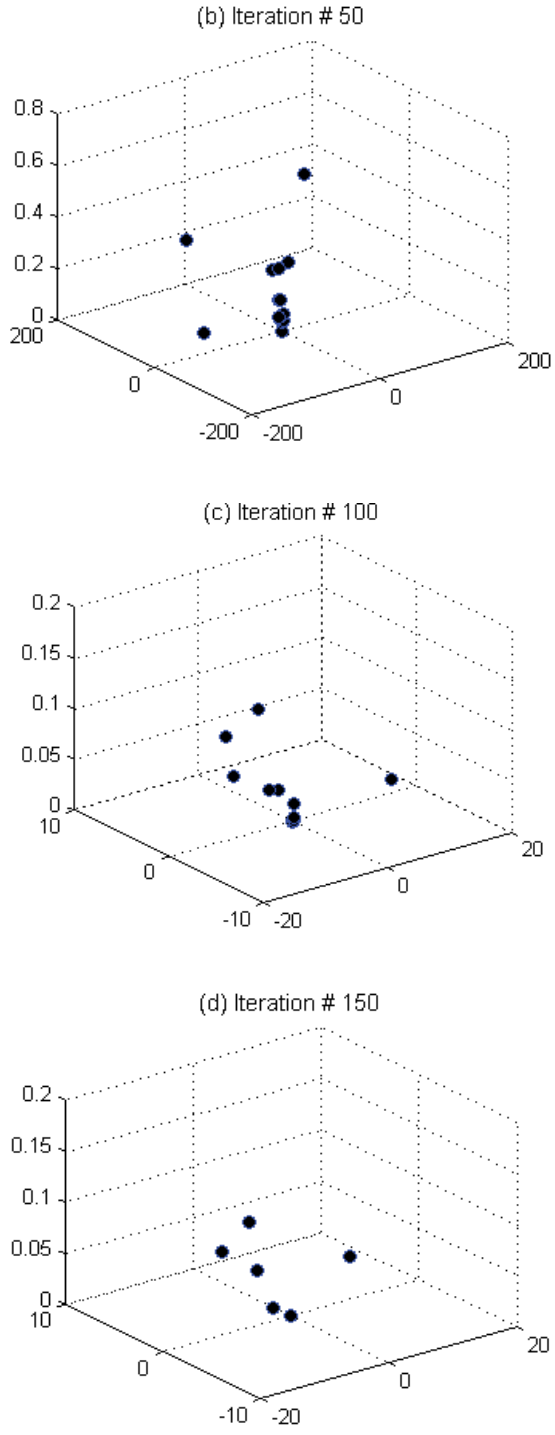


Fig. 6. 3D pictures (a) – (d) of CAIWS-D for 2-dimensional Griewank problem with 20 particles

VI. CONCLUSIONS

In this paper, the performance of the PSO algorithm with two (2) adaptive chaotic inertia weights based on chaotic movement and swarm success rate of particles were proposed and investigated. Their results and performances were extensively compared with those of five (5) existing inertia

weight strategies (RIW, LDIW, CRIW, CDIW and the inertia weight based on decreasing exponential function) recorded in literature, through experimental studies using some nonlinear problems well studied in the literature. Experiment results show that the two proposed inertia weight strategies, CAIWS-R and CAIWS-D, can further improve the performance PSO algorithm in terms of convergence speed and accuracy as well as global search ability because of their chaotic characteristics and adaptive nature provided by the swarm success rate which helped in providing information about the state of the swarm in the search space to adjust the value of inertia weight in each iteration appropriately for better results.

The swarm success rate could be a very useful tool for enhancing the performances of any swarm-based optimization algorithms as a result of the useful information about the particles in the search space it provides, thus its potentiality should be further explored.

Future research will focus on improving on the global search ability, stability and robustness of the proposed methods and then be subjected to comparisons with other adaptive inertia weight strategies. They shall also be applied to solve problems with higher dimensionality and some real-world problems.

ACKNOWLEDGMENT

Our thanks to the College of Agriculture, Engineering and Science, University of Kwazulu-Natal, South Africa for their support towards this work through financial bursary. We also appreciate the reviewers of the original manuscript for their time and efforts, which has made this paper what it is.

REFERENCES

[1] A. Nickabadi , M.M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," Applied soft computing on, vol. 11, pp. 3658-3670, 2011.

[2] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," Swarm Intelligence, vol. 1, pp. 33–57, 2007.

[3] Y. Shi, and R. C. Eberhart, "A modified particle swarm optimizer," Proceedings of the IEEE international conference on evolutionary computation, pp. 69–73, 1998.

[4] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural Exponential Inertia Weight Strategy in Particle Swarm Optimization," Proceedings of the 6th World Congress on Intelligent Control and Automation, June 21 - 23, Dalian, China, 2006.

[5] Y. Feng1, G-F Teng1, A-X Wang1, and Y-M Yao, "Chaotic Inertia Weight in Particle Swarm Optimization," IEEE, 2007.

[6] J. Xin, G. Chen, and Y. Hai , "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," International Joint Conference on Computational Sciences and Optimization, 2009.

[7] N. Iwasaki, K. Yasuda, and G. Ueno, "Dynamic parameter tuning of particle swarm optimization," IEEJ Transactions on electrical and electronic engineering, vol. 1, pp. 353–363, 2006.

[8] "Computational swarm intelligence," part IV, chapter 16, online at http://www.scribd.com/doc/73337453/13/Basic-PSO-Parameters.

[9] R.C. Eberhart, and Y. Shi., "Tracking and optimizing dynamic systems with particle swarms," Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 94–100, 2002.

[10] Y.H. Shi, and R.C. Eberhart, "Fuzzy adaptive particle swarm optimization," Congress on evolutionary computation, Korea, 2001

[11] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, "New particle swarm optimizer with sigmoid increasing inertia weight," International Journal of Computer Science and Security, vol. 1, no. 2, p. 35, 2007.

[12] Y. Gao, X. An, and J. Liu., "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation", International Conference on Computational Intelligence and Security, IEEE, vol. 1, pp. 61–65, 2008.

[13] G. Chen, X. Huang, J. Jia, and Z. Min., "Natural exponential inertia weight strategy in particle swarm optimization", The Sixth World Congress on Intelligent Control and Automation, IEEE, vol. 1, pp. 3672–3675, 2006.

[14] Y. Feng, G.F. Teng, A.X. Wang, and Y.M. Yao., "Chaotic Inertia Weight in Particle Swarm Optimization," Second International Conference on Innovative Computing, Information and Control, IEEE, p. 475, 2008.

[15] H.R. Li and Y.L. Gao., "Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation", Second International Conference on Information and Computing Science, IEEE, pp. 66–69, 2009.

[16] K. Kentzoglanakis, and M. Poole., "Particle swarm optimization with an oscillating inertia weight", Proceedings of the 11th Annual conference on Genetic and evolutionary computation, ACM, pp. 1749–1750, 2009.

[17] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization", Proceedings of IEEE international conference on neural networks, Perth, Australia,vol. 4, pp. 1942–1948, 1995.

[18] J. Ememipour, M.M.S. Nejad, M.M. Ebadzadeh, and J. Rezanejad, "Introduce a new inertia weight for particle swarm optimization", The Fourth International Conference on Computer Sciences and Convergence Information Technology, pp. 1650-1653. IEEE, 2009.

[19] G.I. Evers, "An automatic regrouping mechanism to deal with stagnation in particle swarm optimization," MSc. Thesis, Graduate school of the University of Texas-Pan American, May, 2009.

[20] Y. Gao, X. An, and J. Liu., "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation," International Conference on Computational Intelligence and Security, vol. 1, pp. 61–65. IEEE, 2008.

[21] J. Sun, C-H. Lai, and X-J Wu, "Particle swarm optimization: classical and quantum perspectives", CRC press, New York, 2011.

An Improved Particle Swarm Optimizer based on Swarm Success Rate for Global Optimization Problems

Journal:	*Journal of Experimental & Theoretical Artificial Intelligence*
Manuscript ID:	Draft
Manuscript Type:	Original Article
Keywords:	Particle swarm optimization, Global optimization, Unconstrained Optimization, Inertia weights, Success rate

An Improved Particle Swarm Optimizer based on Swarm Success Rate for Global Optimization Problems

Abstract

Inertia weight is one of the control parameters that influence the performance of Particle Swarm Optimization (PSO) in course of solving global optimization problems, by striking a balance between exploration and exploitation. Among many inertia weight strategies that have been proposed in literature include Chaotic descending inertia weight (CDIW) and chaotic random inertia weight (CRIW). These two strategies have been claimed to perform better than Linear descending inertia weight (LDIW) and Random inertia weight (RIW) PSO variants respectively. Despite these successes, a closer look at their results reveals that the common problem of premature convergence associated with PSO algorithm still lingers. Motivated by the better performances of CDIW and CRIW, this paper proposed two new inertia weight strategies namely: Swarm success rate descending inertia weight (SSRDIW) and Swarm success rate random inertia weight (SSRRIW). These two strategies use swarm success rate as feedback parameter. Efforts were made using the proposed inertia weight strategies with PSO to further improve the effectiveness of the algorithm in terms of convergence speed, global search ability and increased solution accuracy. The proposed PSO variants, $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ were validated using several benchmark unconstrained global optimization test problems and their performances compared with LDIW-PSO, CDIW-PSO, RIW-PSO, CRIW-PSO, and some other existing PSO variants. Empirical results showed that the proposed variants are more efficient.

Keywords: Particle swarm optimization, Success rate, chaos, Inertia weights, Global optimization

1. Introduction

Generally, optimization problems involve how to select the best course of action among many others, given some restrictions. Optimization problems typically have three fundamental elements – objective function, decision variables, and constraints. Objective function is (in many cases) a single numerical quantity that is to be optimized (maximized or minimized). Decision variables are quantities whose values can be manipulated in order to optimize the objective. Constraints are restrictions on the values that the decision variables can take. A global optimization problem can be generally represented in the following way:

$$\text{Optimize } f(\vec{x})$$
$$\text{subject to } g_j(\vec{x}) \leq 0$$

where $\vec{x} = (x_1, x_2, \ldots, x_n)$ is the decision variable in $\Re^n$, f:$\Re^n \to \Re$ is the objective function and g_j: $\Re^n \to \Re$, j *=1, 2, …, m* the constraint functions.

The goal of any optimization problem is to maximize or minimize an objective function. Solution $\vec{x}^*$ is a global minimizer of $f(\vec{x})$ if and only if $f(\vec{x}^*) \leq f(\vec{x})$ for all $\vec{x}^*$ in the domain of $f(\vec{x})$. But it is global maximizer of $f(\vec{x})$ if and only if $f(\vec{x}^*) \geq f(\vec{x})$ for all $\vec{x}^*$ in the domain of $f(\vec{x})$. Optimization problems are often multi-modal (non-convex); that is, they possess multiple good solutions and proffering solutions to such problems is the subject matter of global optimization. The proffered solutions could all be globally good (same cost function value) or there could be a mix of globally good and locally good solutions.

Swarm intelligence is one of the classes of nature-inspired metaheuristics that has been used to provide (near) optimal solutions to many complex optimization problems, over the years. The goal of swarm intelligence is the design of intelligent multi-agent systems by taking inspiration from the collective behavior of social organisms. A popular member of swarm intelligence metaheuristcs is PSO. The notion of PSO originated from the

behaviour of a group of birds which coordinates itself, with some degree of randomness, in order to achieve an objective. The idea of PSO was transferred to optimization by Eberhart and Kennedy in 1995 [3], where each particle (bird) uses its personal experience and that of its neighbours to decide on its own movement from one point to the other, resulting to adaptive swarm emergent behaviour.

Many variants of PSO that exist in literature include [2, 5-7, 9, 12-15, 21, 25]. These variants emanated as a result of the desires by researchers to improve on the performance of PSO technique. Among these variants are LDIW-PSO [22-24], RIW-PSO [4], CDIW-PSO and CRIW-PSO [5], dynamic adaptive PSO (DAPSO) [21] and natural exponential (base e) PSOs (e_1-PSO and e_2-PSO) [2]. In this paper, some efforts were made using the proposed inertia weight strategies to further improve the effectiveness of PSO technique in terms of convergence speed, global search ability and increased solution accuracy. The paper focuses on the inertia weight strategy of PSO to study the effect of swarm success rate as feedback parameter compared to non-feedback chaotic values in the inertia weight formula. Empirical results from numerical experiments performed showed that the proposed PSO variants are very efficient.

The remaining part of this paper is organized in five major sections. Section 2 summarized the inertia weight PSO technique. Section 3 reviewed the PSO variants that were experimentally compared with the proposed variants. Section 4 described the proposed PSO variants while the numerical simulations were carried out in Section 5. Conclusion of the paper is in Section 6.

2. Inertia weight PSO

PSO technique is population-based, adaptive and stochastic in nature. It has a wide range of applications in different fields including economics, engineering, industry, biology and many other complex real world optimization problems [10, 16, 19]. In PSO, a swarm of particles (set of solutions) is randomly positioned in the search space and the quality of each particle is determined by the value of the objective function associated with the problem being optimized. Each particle knows its own best solution and the best solution of the whole swarm and a single population is often maintained but is adjusted in response to new discoveries about the solution space.

To implement PSO involves manipulating about eight different parameters which are particle swarm size, problem dimensionality, particle velocity, inertia weight, particle velocity limits, cognitive learning rate, social learning rate and the random factors. However, the number of parameters needed depends on the PSO variants being implemented. These parameters collectively help the algorithm in the course of optimizing a given problem. Among these parameters, the inertia weight (ω) have attracted much attentions of researchers because of the common belief that it helps in balancing the local and global search of PSO algorithm during the optimization process [20, 22, 26]. Despite the significant and prominent role of inertia weight, it needs the support of other parameters to function effectively [8, 26].

The inertia weight PSO consists of three steps, generating initial positions and velocities for particles, updating the velocities of particles and updating the positions of particles. Each particle in the swarm is made up of two major n-dimension vectors, $X_i = (x_{i1}, \ldots, x_{in})$ represents the position and $V_i = (v_{i1}, \ldots, v_{in})$ represent the velocity of particle i. When the technique is being implemented, the particles move around while adjusting their velocities and positions according to equations (1) and (2), in the search space in search for optimum solution to the problem being optimized.

$$V_i^{k+1} = \omega V_i^k + c_1 r_1 (PB_i^k - X_i^k) + c_2 r_2 (GB^k - X_i^k) \quad (1)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (2)$$

where, c_1 and c_2 are acceleration constants also known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of PB (particle's previous best) and GB (global best); r_1 and r_2 are random numbers between 0 and 1; k is iteration index and ω is inertia weight. The original PSO algorithm [3] uses the values of 1.0, 2.0 and 2.0 for ω, c_1 and c_2 respectively. The positions of the particles are controlled to be within the solution search space while their velocities are clamped within some specified maximum velocity bounds.

3. A review of the PSO variants adopted for comparison

In this section, all the PSO variants considered for comparison with the variants proposed in this paper are reviewed and discussed. These variants are LDIW-PSO, CDIW-PSO, RIW-PSO, CRIW-PSO, e_1-PSO, e_2-PSO and DAPSO.

3.1. Linear decreasing inertia weight PSO (LDIW-PSO)

LDIW-PSO is a variant of PSO which implements the linear descending (or decreasing) inertia weight strategy. The introduction of linearly decreasing inertia weight into the inertia weight PSO greatly improved the algorithm. This was ascertained through experimental studies by [23, 24]. In this variant, the inertia weight starts with some large initial value and then linearly decreases to some smaller final value with the belief that a large inertia weight facilitates a global search while a small inertia weight facilitates a local search. The commonly used initial and final values are 0.9 and 0.4 [2, 5, 15]. However, there are cases where values other than 0.9 or 0.4 are used [12, 13, 21]. With these values, the inertia weight could be seen as the fluidity of the medium in which a particle travels [20]; high initial value makes particles travel in a low viscosity medium, which favours exploration while lower inertia value makes the particle moves in a high viscosity medium favouring exploitation. However, using the linearly decreasing inertia weight makes PSO become victim of premature convergence, despite its quick convergence towards the optimal positions at the beginning [23]. Many attempts have been made to improvement on LDIW-PSO [4, 5, 9, 25, 26]. Equation (3) shows the LDIW strategy.

$$\omega_t = (\omega_{start} - \omega_{end})\left(\frac{T_{max} - t}{T_{max}}\right) + \omega_{end} \tag{3}$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, t is the current iteration number, T_{max} is the maximum iteration number and $\omega_t \in [0,1]$ is the inertia weight value in the t^{th} iteration.

3.2. Chaotic descending inertia weight PSO (CDIW-PSO)

Utilizing the merits of chaotic optimization, CDIW-PSO was proposed by [5] based on logistic mapping in equation (4). Chaos is a nonlinear dynamic system which is sensitive to the initial value. It has the characteristic of ergodicity and stochastic property. The goal was to address the problem of premature convergence associated with LDIW-PSO. Equation (5) represents the chaotic descending inertia weight.

$$z_{k+1} = \mu \times z_k \times (1 - z_k) \tag{4}$$

where $\mu = 4$ and z_k is the k^{th} chaotic number. The map generates values between 0 and 1, provided that the initial value $z_0 \in (0,1)$ and that $z_0 \notin (0.0, 0.25, 0.5, 0.75, 1.0)$.

$$\omega_t = (\omega_{start} - \omega_{end})\left(\frac{T_{max} - t}{T_{max}}\right) + \omega_{end} \times z_{k+1} \tag{5}$$

where ω_{start} and ω_{end} are as defined above. CDIW-PSO demonstrated better convergence precision, quick convergence velocity, and better global search ability compared with LDIW-PSO [5].

3.3. Random inertia weight PSO (RIW-PSO)

In [4], randomness was introduced into the inertia weight strategy in PSO. Using particle swarms to track and optimize dynamic systems, a new way of calculating the inertia weight value was proposed as shown in equation (6). The formula produces a number randomly varying between 0.5 and 1.0, with a mean value of 0.75 while c_1 and $c_2 = 1.494$.

$$\omega_t = 0.5 + \frac{rand()}{2} \tag{6}$$

As a result of the difficulty in predicting whether exploration (a larger inertia weight value) or exploitation (a smaller inertia weight) will be better at any given time in tracking a nonlinear dynamic system, the strategy in equation (6) was introduced to address the inefficiency of linearly decreasing inertia weight, which decreases from 0.9 to 0.4 during a run, in handling such a problem.

3.4. Chaotic random inertia weight PSO (CRIW-PSO)

The chaotic random inertia weight (CRIW) was proposed in [5] as shown in equation (7). The aim was to improve on the random inertia weight in equation (6) using logistic map in equation (4), to avoid getting into local optimum in searching process by utilizing the merits of chaotic optimization.

$$\omega_t = \frac{rand()}{2} + 0.5 \times z_{k+1} \quad (7)$$

where *rand()* is a uniform random number in [0,1] The results in [5] show that the PSO had preferable convergence precision, quick convergence velocity, and better global search ability. This is because, due to non-repetition of chaos the algorithm could carry out overall searches at higher speed and diversify the particles and improves the algorithm's performance in preventing premature convergence too quickly to local minima compared with RIW which have no chaos characteristics.

3.5. Dynamic adaptive particle swarm optimization (DAPSO)

This variant was proposed by [21] to solve the PSO premature convergence problem associated with typical multi-peak, high dimensional function optimization problems and improve its global optimum convergence speed. So as to achieve this goal, a dynamic adaptive strategy was introduced into the variant to adjust the inertia weight value based on the current swarm diversity. Experimental results showed evidences that DAPSO performed better than LDIW-PSO. The inertia weight formula that was used is represented in equation (8).

$$\omega_t = \omega_{min} + (\omega_{max} - \omega_{min}) \times F_t \times \varphi_t \quad (8)$$

where $\omega_{\min}$ and $\omega_{\max}$ are the minimum and maximum inertia weight values, t is the current number of iterations, the diversity function F_t and adjustment function φ_t, both in the t^{th} iteration are represented in equations (9) and (10) respectively.

$$F_t = 1 - \frac{2}{\pi} arc\tan(E) \quad (9)$$

where E is the group fitness as shown in equation (11).

$$\varphi_t = e^{(-t^2/(2\sigma^2))} \quad (10)$$

where, $\sigma = \frac{T}{3}$ and T is the total number of iterations.

$$E = \frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - f_{avg}\right)^2 \quad (11)$$

Where N is the swarm size, $f(x_i)$ is the fitness of particle i and f_{avg} represented in equation (12) is the current average fitness of the swarm.

$$f_{avg} = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \quad (12)$$

3.6. Natural exponential inertia weight PSO

Based on the idea of decreasing inertia weight strategy, [2] proposed two inertia weight strategies of natural exponential functions, e_1-PSO and e_2-PSO represented by equations (13) and (14) respectively. Based on the experimental settings in [2], these inertia weight strategies were proved to converge faster than LDIW-PSO during the early stage of the search process. Besides, they were also claimed to have performed better in most of the continuous optimization problems that were solved.

$$\omega_t = \omega_{min} + (\omega_{max} - \omega_{min}) e^{-\left[\frac{t}{\left(\frac{MAXITER}{10}\right)}\right]} \quad (13)$$

$$\omega_t = \omega_{min} + (\omega_{max} - \omega_{min}) e^{-\left[\frac{t}{\left(\frac{MAXITER}{4}\right)}\right]^2} \quad (14)$$

where, t is the current iteration number and *MAXITER* is the maximum allowed number of iterations.

3.7. Discussions

The inertia weight PSO is generally not difficult to implement and it is fast in convergence. Looking at the variants described above, they all tried to address the problem of getting stuck in local optima associated with PSO. They are basically of two groups – *randomly* and *linearly* inclined, though with some infusion of chaotic

and swarm variance into some of them for enhancement. For the randomly-inclined inertia weight strategies, they are more or less purely controlled by randomly (or chaotic) generated numbers while the linearly-inclined strategies are controlled by the iteration number of the algorithm; and none of them has any information about the state of the swarm in the search space which could influence the nature of search for optimal solution by the swarm. Besides, the linearly directed strategies have the initial and final values of the inertia weight fixed, thereby ruling out the flexibility of obtaining some lower or higher values for the inertia weight that could help the algorithm obtain some good optima results. Therefore, it is of utmost importance that some means be devised to help reaslise the state of the swarm in the search space as well as create some flexibility in either of the limits of the inertia weight with the belief that this could help the algorithm obtain some better results.

4. Proposed PSO variants

One of the major goals of this work is to enhance the performance of the inertia weight PSO. In achieving this, the following 3 definitions are given:

***Definition* 1:** Given that the current position of particle *i* at iteration *t* is $Pbest_t^i$ and the function to be optimized as *f( )*. The success of particle *i* at iteration *t*, in a minimization problem, is defined in equation (15).

$$succ_t^i = \begin{cases} 1 & ,if\ f(Pbest_t^i) < f(Pbest_{t-1}^i) \\ 0 & ,if\ f(Pbest_t^i) \geq f(Pbest_{t-1}^i) \end{cases} \quad (15)$$

***Definition* 2:** Given a swarm of particles of size *n* and the success of each particle at iteration *t* to be $succ_t^i$. The swarm success rate (*ssr*) at iteration *t* is defined as shown in equation (16).

$$ssr_t = \frac{\sum_i^n succ_t^i}{n} \quad (16)$$

From *definition 2*, $ssr_t \in [0,1]$. Where no particle succeeds to improve its fitness, ssr_t is set to *0*; where all the particles succeed in improving their individual fitness, it is set to *1*. Therefore ssr_t reflects the state of the swarm and can serve as a feedback parameter to the PSO algorithm to determine the inertia weight at the t^{th} iteration.

***Definition* 3:** Given ω_{min} and ω_{max} as the minimum and maximum inertia weight values, T_{max} as the maximum iteration number and *t* as the current iteration of the algorithm, the two proposed inertia weight strategies $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$, are defined by equations (17) and (18) respectively.

$$\omega_t = (\omega_{start} - \omega_{end})\left(\frac{T_{max} - t}{T_{max}}\right) + \omega_{end} \times ssr_{t-1} \quad (17)$$

$$\omega_t = 0.5 \times rand() + 0.5 \times ssr_{t-1} \quad (18)$$

where, ssr_{t-1} is the swarm success rate at the previous iteration which adaptively determine the lower limit of the range of inertia weight values. This was made so because, having a fixed final inertia weight value (i.e., ω_{end} and 0.5) could limit the flexibility of the inertia weight strategy obtaining some possible lower or higher values around ω_{end} and 0.5 that could contribute to its effectiveness. Besides, the algorithms could perform better if there is a way the state of the swarm in the search space could be fed back into the system. Thus, ssr_{t-1} as a feedback parameter could help realize the state of the swarm in the search space and hence adjust the value of inertia weight at each iteration appropriately for better results. Among the goals of this paper is to further improve on the performances of LDIW-PSO and RIW-PSO using this idea and then verify the superiorities of the proposed variants in comparison with CDIW-PSO, CRIW and some other existing PSO variants. Figure 1 shows the algorithm for the proposed variants.

5. Numerical simulations

To validate the performance of the proposed PSO variants, four different experiments were performed for the purpose of detailed comparison of $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ with seven other different PSO variants namely, LDIW-PSO, RIW-PSO, CDIW-PSO, CRIW-PSO, DAPSO, e_1-PSO and e_2-PSO. Different experiments, relative to the competing PSO variants, used different set of test problems which were also used to test the proposed variants. The program was developed with Window-based Microsoft Visual C# programming language, running on a system with a 2.0GHz Intel Pentium dual-core processor, 2.0GB of RAM.

5.1. Benchmark problems

Different test problems with varied difficulties were used to verify the performance of the proposed variants. Descriptions of all the test problems, with some additional information found in [11, 17, 18, 27, 28], are given in Appendix I. Shown in Tables 1 and 2 are the names, search ranges, optimum values, dimensions and characteristics (US – unimodal separable, UN – unimodal non-separable, MS – multimodal separable, MN – multimodal non-separable) of the respective test problems.

5.2. Settings of the experiment

The settings of the different experiments performed for the comparisons are described below one after the other. In experiments 1 – 3, $SSRDIW_{PSO}$ and $SSRDIW_{PSO}$ were subjected to the same settings of its competitors as recorded in literature [2, 5, 21]. All experiments were made to run for the maximum number of iterations which serves as the stopping criteria for all the algorithms. In all the experiments in this paper, the initial positions of particles were generated using uniform random number generator.

5.2.1. Experiment 1

In this experiment $SSRDIW_{PSO}$ and $SSRDIW_{PSO}$ were respectively compared with the two PSO variants, CDIW-PSO and CRIW-PSO adopted from [5]. The test problems used were *Griewank*, *Rastrigin*, *Rosenbrock*, *Schaffer's f6* and *Sphere* problems as shown in Table 1. The problems dimensions and stopping criteria are stated in Table 3. The maximum numbers of iterations was set to 1500 with swarm size of 20 and the experiment was repeated 500 times for each test problem. All these settings were adopted from [5]. The goal of this experiment was to verify whether the proposed variants are more efficient than their competitors.

5.2.2. Experiment 2

In this experiment, another settings used for e_1-PSO and e_2-PSO in [2] were adopted. Apart from the parameter V_{max} set to be $0.05X_{max}$ for $SSRDIW_{PSO}$ and $SSRDIW_{PSO}$, all other settings remain the same. The performances of the proposed variants were compared with that of e_1-PSO and e_2-PSO. The test problems used were *Griewank*, *Rastrigin*, *Rosenbrock* and *Sphere* problems as shown in Table 1. The problems dimensions and stopping criteria are stated in Table 4. The maximum numbers of iterations was set to 3000 with swarm size of 30 and the experiment was repeated 50 times for each of the test problems. These settings were adopted from [2].

5.2.3. Experiment 3

In this experiment, $SSRDIW_{PSO}$ and $SSRDIW_{PSO}$ were further tested by subjecting them to different settings used for DAPSO in [21]. The test problems used were *Ackley*, *Griewank* and *Rastrigin* problems. The problems dimensions and search ranges are stated in Table 5. The maximum numbers of iterations was set to 3000 with swarm size of 30 and the experiment was repeated 50 times for each of the test problems. All settings were adopted from [21].

5.2.4. Experiment 4

After the preceding experiments, $SSRDIW_{PSO}$, $SSRDIW_{PSO}$, LDIW-PSO, RIW-PSO, CDIW-PSO and CRIW-PSO were subjected to the same experimental settings to optimize 12 test problems. Subjecting the variants to the same experimental settings gives each of them equal opportunity of performance.

A common platform of test problems, search ranges, dimensions and success criteria, all shown in Tables 7 and 8, was set for all the competing variants to test their respective performance. Table 7 contains 6 high-scaled benchmark test problems while Table 8 contains 6 low-scaled benchmark test problems. The swarm size was set to 20 and 30; maximum allowed number of iterations was 1000, 3000 and 5000 for three respective different problem dimensions of 10, 30 and 50. A maximum iteration of 1000 was used for the low-scaled problems. The experiment was repeated 100 times for each of the test problems. In the experiment, $c_1 = c_2 = 2.0$, $\omega_{max} = 0.9$, $\omega_{min} = 0.4$.

For the purpose of fair comparison, an experiment was conducted to obtain a suitable setting for V_{min} and V_{max} for CDIW-PSO, CRIW-PSO, LDIW-PSO and RIW-PSO. The experiment tested these four variants with $V_{min} = X_{min}$ and $V_{max} = X_{max}$ on the one hand and $V_{min} = 0.05X_{min}$ and $V_{max} = 0.05X_{max}$ on the other. Ackley, *Griewank*, *Rastrigin* and *Rosenbrock* problems were used with 20 particles and maximum iteration of 3000. The experiment was repeated 100 times. Presented in Table 6 are the results obtained from the experiment, showing the mean best fitness for all the problems as obtained by the four PSO variants.

From the results in Table 6, it was discovered that CDIW-PSO, CRIW-PSO, LDIW-PSO and RIW-PSO performed very well using $V_{min} = 0.05X_{min}$ and $V_{max} = 0.05X_{max}$ compared with $V_{min} = X_{min}$ and $V_{max} = X_{max}$. Therefore, the particle velocity bounds that were used in this experiment 4 for all the competing variants was $V_{min} = 0.05X_{min}$ and $V_{max} = 0.05X_{max}$.

5.3. Comparative study and Discussions

In this sub-section results obtained from all the experiments are presented and discussed. The results were compared based on the final accuracy of the mean best solutions (Mean fitness), convergence speed, standard deviation (Std. Dev.) and success rate (SR).The mean best fitness (solution) is a measure of the precision that the algorithm can get within given iterations while the standard deviation is a measure of the algorithm's stability and robustness and success rate is the rate of the optimum fitness result in the criterion range experimenting a number times independently.

Presented in Tables 9 – 31 are the results obtained for all the experiments. The results for all the competing PSO variants for experiments 1 – 3 were obtained from the respective referenced papers and they are presented here the way they were recorded. Thus, the recording of the results for $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ were patterned after them. In each of the tables, bold values represent the best results.

5.3.1. Results for Experiment 1

The results for CDIW-PSO were adopted from [5]. The results in Table 9 clearly reveal a great difference in performance between $SSRDIW_{PSO}$ and CDIW-PSO. The results are compared based on the final accuracy of the averaged best solutions and success rate (SR%). In all the test problems, the result indicates that $SSRDIW_{PSO}$ can get better optimum fitness results, showing better convergence precision. Besides, $SSRDIW_{PSO}$ has better global search ability and could easily get out of local optima than CDIW-PSO.

Table 10 compares the performances $SSRRIW_{PSO}$ and CRIW-PSO. The results for CRIW-PSO were adopted from [5]. The results are also compared based on the final accuracy of the averaged best solutions and success rate (SR%). The results show that in all the test problems, $SSRRIW_{PSO}$ can get better optimum fitness results, showing better convergence precision. However, there are some differences in their global search abilities in favour of CRIW-PSO in *Griewank* and *Rosenbrock* problems with slight differences. But $SSRRIW_{PSO}$ has better global search ability in *Rastrigin* and *Schaffer's f6* problems with high differences.

5.3.2. Results for Experiment 2

In Tables 11 and 12, $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ are compared together with e_1-PSO and e_2-PSO based on their final accuracies of the averaged best solutions and number of trials that successfully reached the stopping criteria. The results for e_1-PSO and e_2-PSO were adopted from [2]. In all the test problems, $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ outperformed e_1-PSO as well as e_2-PSO. They got better optimum fitness results, demonstrated better convergence precision global search ability showing that they could easily get out of local optima than their competitors. However, e_1-PSO was three trials higher in reaching the stopping criteria for *Rosenbrock* problem.

5.3.3. Results for Experiment 3

The results for DAPSO were obtained from [21]. As shown in Table 13, these results were compared with those of $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ based on the final accuracy of the respective mean best solutions across the different problems dimensions. In all the problems and dimensions, $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ outperformed DAPSO in getting better fitness quality and precision. This is a clear indication that in both global search ability and not easily getting trapped in local optima, $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$ are superior to DAPSO. Generally, $SSRRIW_{PSO}$ performed better than $SSRDIW_{PSO}$.

5.3.4. Results for Experiment 4

In this sub-section an in-depth empirical and comparison studies were carried out based on the obtained results from experiment 4, to find which variants among $SSRDIW_{PSO}$, LDIW-PSO and CDIW-PSO on the one hand and $SSRRIW_{PSO}$, RIW-PSO and CRIW-PSO on the other hand, could obtain outstanding results with the intent of validating the performances of the two proposed inertia weight strategies when used with the inertia weight PSO algorithm.

5.3.4.1. Comparison of $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO using the high-scaled test problems

Tables 14 – 19 show all the results of all the variants for six high-scaled problems using two different swarm sizes (20 and 30) and three different problems dimensions (10, 30 and 50). When the dimension was set to 10 for all the problems (Tables 14a, 14b, 15a and 15b), $SSRDIW_{PSO}$ obtained better optimal results for all the problems, except *Rosenbrock* when swarm size was 20 and *Rastrigin* when swarm size was 30, that CDIW-PSO had better optimal results. Though, LDIW-PSO was able to get good results but could not perform better that the others.

When the dimension was set to 30 for all the problems (Tables 16a, 16b, 17a and 17b), $SSRDIW_{PSO}$ also obtained better optimal results for all the problems, except *Ackley* when swarm size was 20 and *Rosenbrock* when swarm size was 30, where CDIW-PSO had better optimal results and *Rastrigin* when swarm size was 30, where LDIW-PSO had better optimal result.

With the dimension set to 50 for all the problems (Tables 18a, 18b, 19a and 19b), $SSRDIW_{PSO}$ could only obtain better optimal results in *Rastrigin*, *Schwefel P2.22* and *Sphere* problems while CDIW-PSO performed better in *Ackley* and *Griewank* problems and LDIW-PSO in *Rosenbrock* when swarm size was 20. When swarm size was 30, $SSRDIW_{PSO}$ took lead in *Ackley*, *Schwefel P2.22* and *Sphere* problems while CDIW-PSO took lead in *Rastrigin* and *Rosenbrock*, but LDIW-PSO in *Griewank*.

Table 20 gives the summarized of the average performance ranking in terms of mean best fitness for LDIW-PSO, CDIW-PSO and $SSRDIW_{PSO}$ across all the swarm sizes and problems dimensions for the test problems. From the table, $SSRDIW_{PSO}$ had the best overall performance compared with its competitors when the problems dimensions were set to 10 and 30, but CRIW-PSO slightly performed better than $SSRDIW_{PSO}$ when the problems dimension was 50. RIW-PSO had the least performance.

Figure 2 below shows the convergence curves and the searching behaviour of $SSRDIW_{PSO}$ compared with CDIW-PSO and LDIW-PSO in all the six high-scaled test problems with swarm size of 20 and a dimension of 30. Apart from *Ackley* problem, $SSRDIW_{PSO}$ was consistent in demonstrating better convergence and obtaining better quality results than other variants in all other test problems.

5.3.4.2. Comparison of $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO using the high-scaled test problems

Tables 21 – 26 show all results for the variants for six scalable problems using two different swarm sizes and three different problems dimensions. When the dimension was set to 10 for all the problems (Tables 21a, 21b, 22a and 22b), $SSRRIW_{PSO}$ obtained better optimal results for all the problems, except *Griewank* and *Rosenbrock* when swarm size was 20 and *Rastrigin* when swarm size was 30, that CRIW-PSO had better optimal results.

When the dimension was set to 30 for all the problems (Tables 23a, 23b, 24a and 24b), $SSRRIW_{PSO}$ also obtained better optimal results for all the problems, except *Ackley* when swarm size was 20 and *Rastrigin* when swarm size was 30, where CRIW-PSO had better optimal results.

With the dimension set to 50 for all the problems (Tables 25a, 25b, 26a and 26b), $SSRRIW_{PSO}$ still obtained better optimal results for all the problems, except *Ackley* when swarm size was 20 and 30 where CRIW-PSO performed better.

Table 27 summarizes the average performance ranking in terms of mean best fitness for RIW-PSO, CRIW-PSO and $SSRRIW_{PSO}$ across all the swarm sizes and problems dimensions for the test problems. From the table, $SSRRIW_{PSO}$ had the best overall performance compared with its competitors in all the test problems, followed by CRIW-PSO. RIW-PSO performed the least.

Figure 3 shows the convergence curves and the searching behaviour of $SSRRIW_{PSO}$ compared with CRIW-PSO and RIW-PSO in all the six high-scaled test problems with swarm size of 20 and a dimension of 30. Apart from *Ackley* problem, $SSRRIW_{PSO}$ was consistent in demonstrating better convergence and obtaining better quality results than other variants in all other test problems.

5.3.4.3. Comparison of $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO using the low-scaled test problems

Tables 28 and 29 show all results for the variants for six low-scaled problems using two different swarm sizes. When the swarm size was set to both 20 and 30 for all the problems (Tables 28a, 28b, 29a and 29b), all the variants almost performed equally except in *Michalewicz* where $SSRDIW_{PSO}$ obtained better optimal result than others and *Schaffer's f6* where LDIW-PSO had better optimal result.

Figure 4 shows the convergence curves and the searching behaviour of $SSRDIW_{PSO}$ compared with CDIW-PSO and LDIW-PSO in 4 of the six low-scaled test problems with swarm size of 20. $SSRDIW_{PSO}$ was consistent in its speed of convergence in *Booth* to obtain the optimum earlier than others. It also performed better in *Michalewicz* than others. However, LDIW-PSO had a better convergence speed at the later stage with better search ability to obtain high quality result in *Schaffer's f6* than others but of the same convergence speed with $SSRDIW_{PSO}$ in *Shurbert*. $SSRDIW_{PSO}$ was able to recover from being trapped very close to the optimum.

5.3.4.4. Comparison of $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO using the low-scaled test problems

Tables 30 and 31 show all results for the variants for six low-scaled problems using two different swarm sizes. When the swarm size was set to both 20 and 30, $SSRRIW_{PSO}$ obtained better optimal result in *Michalewicz* while RIW-PSO had better optimal result in *Schaffer's f6*. RIW-PSO and CRIW-PSO had equal performance in *Shubert.*

The convergence curves in Figure 5 provide insight into the searching behaviour of the competing variants in 4 of the six low-scaled test problems with swarm size of 20. It is clearly shown that $SSRRIW_{PSO}$ was consistent in its speed of convergence in *Booth* to obtain the optimum earlier than others. It also performed better in *Michalewicz* than others. However, RIW-PSO had a better convergence speed at the later stage with better search ability to obtain high quality result in *Schaffer's f6* than others but of the same convergence speed with CRIW-PSO in *Shubert*.

6. Conclusion

Motivated by CDIW and CRIW, in this paper two inertia weight strategies, SSRDIW and SSRRIW, have been introduced into the inertia weight PSO thereby leading to proposing two PSO variants, $SSRDIW_{PSO}$ and $SSRDIW_{PSO}$. To verify whether these variants were as efficient as some of the existing PSO variants, the performance of $SSRDIW_{PSO}$ was extensively compared with those of LDIW-PSO and CDIW-PSO while that of $SSRDIW_{PSO}$ with those of RIW-PSO and CRIW-PSO through experimental studies of some nonlinear functions well studied in the literature. From the experiments conducted, results show that SSRDIW and SSRRIW are more efficient and robust than their competitors in high-scaled problems than in low-scaled. The proposed variants were also compared with e_1-PSO, e_2-PSO and DAPSO and it was discovered that they were also more efficient and robust. In all, the proposed inertia weight strategies have greatly improved the robustness, accuracy and convergent speed of the inertial weight PSO. These proposed variants are based on the fact that usually, a uniform random number does not convey much information about the state of the swarm and therefore cannot be assumed to adjust the inertia weight appropriately. The better performances of the proposed variants was as result of the fact that, the swarm success rate which served as a feedback parameter helped in realizing the state of the swarm in the search space and hence adjusted the value of inertia weight in each iteration appropriately for better results. Finally, there is room for further studies on the proposed PSO variants especially in applying them to constrained global optimization problems as well as real-world optimization problems.

References

[1] Cao, B., Shen, X. and Qian, Q. (2010), Application of two-order particle swarm optimization algorithm in image segmentation, in *Computer-Aided Industrial Design & Conceptual Design (CAIDCD), 2010 IEEE 11th International Conference on*, pp. 749-752.

[2] Chen, G. Huang, X. Jia, J. and Min., Z., (2006). Natural exponential Inertia Weight strategy in particle swarm optimization, In Sixth World Congress on Intelligent Control and Automation, WCICA, June 21 - 23, Dalian, China, 1, pp. 3672–3675.

[3] Eberhart, R. C. and Kennedy, J., (1995), A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95.* (Nagoya, Japan, 1995), pp. 39-43.

[4] Eberhart, R. C. and Shi, Y. (2002), Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress* on *Evolutionary Computation,* Korea, 1, pp. 94–100.

[5] Feng, Y. Teng, G.F. Wang, A.X. and Yao., Y.M., (2008), Chaotic Inertia Weight in Particle Swarm Optimization, In Innovative Computing, Information and Control, ICICIC'07. Second International Conference on, pp. 475.

[6] Gao Y.-l. and Duan, Y.-h., (2007), A New Particle Swarm Optimization Algorithm with Random Inertia Weight and Evolution Strategy, In *Computational Intelligence and Security Workshops, 2007. CISW 2007. International Conference on*, pp. 199-203.

[7] Gao, Y. An, X. and Liu.,J., (2008), A Particle Swarm Optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation, In Computational Intelligence and Security, 2008. CIS'08. International Conference on, 1, pp. 61–65.

[8] Iwasaki, N., Yasuda, K., and Ueno G. (2006). Dynamic Parameter Tuning of Particle Swarm Optimization. *Transactions on electrical and electronic engineering, IEEJ Trans*, 1, pp. 353–363

[9] Jianbin Xin, Guimin Chen, Yubao Hai (2009). A Particle Swarm Optimizer with Multi-Stage Linearly-Decreasing Inertia Weight. *International Joint Conference on Computational Sciences and Optimization*

[10] Jing, B., Xueying, Z. and Yueling, G. (2009), Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system, in *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pp. 4754-4759.

[11] Karaboga, D., and Akay B. (2009). A Comparative Study of Artificial Bee Colony Algorithm. *Applied Mathematics and Computation*, 214, pp. 1.0-132.

[12] Kentzoglanakis, K. and Poole., M., (2009), Particle swarm optimization with an oscillating Inertia Weight, In Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 1749–1750.

[13] Li H. R. and Gao.Y. L., (2009), Particle Swarm Optimization Algorithm with Exponent Decreasing Inertia Weight and Stochastic Mutation, In Second International Conference on Information and Computing Science, pp. 66–69.

[14] Liu, H., Su, R., Gao Y., and Xu R., (2009), Improved Particle Swarm Optimization Using Two Novel Parallel Inertia Weights. IEEE Second International Conference on Intelligent Computation Technology and Automation, pp 185-188.

[15] Malik, R.F. Rahman, T.A. Hashim, S.Z.M. and Ngah, R., (2007), New Particle Swarm Optimizer with Sigmoid Increasing Inertia Weight, *International Journal of Computer Science and Security*, 1(2), pp. 35.

[16] Mansour, M. M., Mekhamer, S. F. and El-Kharbawe, N. E. S. (2007), A Modified Particle Swarm Optimizer for the Coordination of Directional Overcurrent Relays, *Power Delivery, IEEE Transactions on,* vol. 22, pp. 1400-1410.

[17] Molga, M., and Smutnicki, C. (2005). Test functions for optimization needs. Available: http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf. [Cited May, 2013].

[18] Montaz A., M. Khompatraporn, C. and Zabinsky, Z. B., (2005), A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems, *J. of Global Optimization*, 31, pp. 635-672.

[19] Ngoc,D. Vo, Schegner, P., and Ongsakul, W. (2011), A newly improved particle swarm optimization for economic dispatch with valve point loading effects, in *Power and Energy Society General Meeting, 2011 IEEE*, pp. 1-8.

[20] Poli R., Kennedy J., Blackwell T. (2007). Particle swarm optimization: An overview. *Swarm Intelligence*. Vol. 1, pp 33–57.

[21] Shen, X., Chi, Z., Yang, J., Chen, C., and Chi, Z. (2010), Particle swarm optimization with dynamic adaptive inertia weight. In *International Conference on Challenges in Environmental Science and Computer Engineering, IEEE*, 287-290.

[22] Shi, Y. H., Eberhart, R. C., (1998), A modified particle swarm optimizer. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, pp. 69-73.

[23] Shi, Y. H., Eberhart, R. C., (1999), Empirical study of particle swarm optimization. IEEE International Conference on Evolutionary Computation, Washington, USA, pp. 1945-1950.

[24] Shi, Y. and Eberhart, R., (1998), Parameter selection in particle swarm optimization, in *Evolutionary Programming VII*. vol. 1447, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., ed: Springer Berlin Heidelberg, pp. 591-600.

[25] Shi, Y. H., and Eberhart, R. C. (2001), Fuzzy adaptive particle swarm optimization. *Congress on evolutionary computation*, Korea.

[26] Engelbrecht, A.P. (2007). *Computational Intelligence : An Introduction.* John Wiley & Sons Inc., USA.

[27] Chetty, S. and Adewumi, A.O. (2013). Three New Stochastic Local Search Algorithms for Continuous Optimization Problems. *Computational Optimization and Applications*, Springer Online first doi: 10.1007/s10589-013-9566-3.

[28] Sawyerr, B.A., Ali, M.M. & Adewumi, A.O. (2011) A Comparative Study of Some Real-Coded Genetic Algorithms for Unconstrained Global Optimization, *Optimization Methods and Software*, 26:6, 945-970, DOI: 10.1080/10556788.2010.491865

Table 1: High-scaled benchmark problems

Test problem	*Ackley*	*Griewank*	*Rastrigin*	*Rosenbrock*	*Schwefel2.22*	*Sphere*
Optimum	0	0	0	0	0	0
Characteristics	MN	MN	MS	UN	UN	US

Table 2: Low-scaled benchmark problems

Test problem	*Booth*	*Esom*	*Michalewicz5*	*Schaffer*	*Shubert*	*Trid6*
Optimum	0	-1	-4.6877	0	-186.73	-50
Characteristics	MS	UN	MS	MN	MN	UN

Table 3: Settings for experiment 1

Test problem	*Griewank*	*Rastrigin*	*Rosenbrock*	*Schaffer's f6*	*Sphere*
Dimension	30	30	30	2	30
Search range	±600	±30	±30	±100	±100
Criteria	0.05	50.0	100.0	0.00001	0.01

Table 4: Settings for experiment 2

Test problem	*Griewank*	*Rastrigin*	*Rosenbrock*	*Sphere*
Dimension	30	30	30	30
Search range	±600	±10	±100	±100
Criteria	0.02	50.0	50.0	10^{-10}

Table 5: Settings for experiment 3

Test problem	*Ackley*	*Griewank*	*Rastrigin*
Dimension	20, 30, 40 and 50 for all problems		
Search range	±32	±600	±5.12

Table 6: Mean best fitness to determine better V_{max} for the PSO variants

Problem	**LDIW-PSO**		**RIW-PSO**		**CDIW-PSO**		**CRIW-PSO**	
	$V_{max} = X_{max}$	$V_{max} = 0.05X_{max}$	$V_{max} = X_{max}$	$V_{max} = 0.05X_{max}$	$V_{max} = X_{max}$	$V_{max} = 0.05X_{max}$	$V_{max} = X_{max}$	$V_{max} = 0.05X_{max}$
Ackley	4.4697e+00	1.8661e-09	7.2254e+00	2.9080e-01	1.6826e+00	2.4425e-14	9.5966e-01	1.0022e-11
Griewank	9.9348e+00	1.4810e-02	1.1114e+01	8.5596e-01	9.0377e+00	1.3234e-02	9.1751e-01	1.4164e-02
Rastrigin	9.5103e+01	3.2638e+01	1.7412e+02	3.3576e+01	7.8947e+01	3.4617e+01	6.2993e+01	3.3573e+01
Rosenbrock	3.0349e+04	2.9721e+01	9.2522e+04	7.6139e+01	1.3009e+04	3.1915e+01	6.7331e+03	3.2548e+01

Table 7: Scalable benchmark problems

Test problem	*Ackley*	*Griewank*	*Rastrigin*	*Rosenbrock*	*Schwefel2.22*	*Sphere*
Search Range	±30	±600	±5.12	±30	±10	±100
Dimension	Different dimensions – 10, 30 and 50 were used for each of the problems					
Criteria	0.001	0.05	50	50	0.001	0.001

Table 8: Non-scalable benchmark problems

Test problem	*Booth*	*Esom*	*Michalewicz5*	*Schaffer*	*Shubert*	*Trid6*
Search Range	±10	±100	[0,π]	±100	±10	±36
Dimension	2	2	5	2	2	6
Criteria	0.00001	-0.999	-4.687	0.00001	-186.729	-49.999

Table 9: Experimental results for $SSRDIW_{PSO}$ compared with CDIW-PSO

	Griewank		Rastrigin		Rosenbrock		Schaffer's f6		Sphere	
	CDIW-PSO	*SSRDIW*$_{PSO}$	*CDIW-PSO*	*SSRDIW*$_{PSO}$	*CDIW-PSO*	*SSRDIW*$_{PSO}$	*CDIW-PSO*	*SSRDIW*$_{PSO}$	*CDIW-PSO*	*SSRDIW*$_{PSO}$
Mean fitness	0.014773	**0.012153**	40.044561	**33.053889**	44.305058	**31.768752**	0.007732	**0.005106**	0.000092	**0.000000**
Std. Dev.	0.002959	0.015943	8.028912	10.700165	8.861012	21.221641	0.001546	0.004846	0.000016	0.000000
SR (%)	96.2	**97.2**	83.6	**92.6**	**99.6**	99.2	22.0	**46.4**	100	100

Table 10: Experimental results for $SSRRIW_{PSO}$ compared with CRIW-PSO

	Griewank		Rastrigin		Rosenbrock		Schaffer's f6		Sphere	
	CRIW-PSO	*SSRRIW*$_{PSO}$	*CRIW-PSO*	*SSRRIW*$_{PSO}$	*CRIW-PSO*	*SSRRIW*$_{PSO}$	*CRIW-PSO*	*SSRRIW*$_{PSO}$	*CRIW-PSO*	*SSRRIW*$_{PSO}$
Mean fitness	0.016616	**0.013559**	40.267957	**31.721714**	37.090110	**33.930754**	0.009211	**0.005615**	0.000087	**0.000000**
Std. Dev.	0.003323	0.016669	8.053591	9.816728	11.618022	24.488409	0.001842	0.005204	0.000017	0.000000
SR (%)	**98.2**	95.4	91.8	**96.0**	**99.4**	98.8	24.4	**43.2**	100	100

Table 11: $SSRDIW_{PSO}$, $SSRRIW_{PSO}$, e_1-PSO and e_2-PSO

	Griewank				Rastrigin			
	e$_1$*-PSO*	*e*$_2$*-PSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$	*e*$_1$*-PSO*	*e*$_2$*-PSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$
Mean fitness	0.0186	0.0101	**0.0080**	0.0112	52.5772	53.3436	52.0344	**49.2309**
Std. Dev.	0.0183	0.0126	**0.0109**	0.0111	11.1903	15.9680	14.0604	**9.8200**
Trials reaching stopping criteria	31/50	39/50	**45/50**	42/50	19/50	17/50	24/50	**25/50**

Table 12: $SSRDIW_{PSO}$, $SSRRIW_{PSO}$, e_1-PSO and e_2-PSO

	Rosenbrock				Sphere			
	e$_1$*-PSO*	*e*$_2$*-PSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$	*e*$_1$*-PSO*	*e*$_2$*-PSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$
Mean fitness	57.3858	70.8313	**28.9155**	32.4599	9.3505e-11	9.3093e-11	**1.5552e-58**	9.1074e-53
Std. Dev.	86.4313	115.9860	**21.7438**	31.4591	5.0294e-12	6.2098e-12	**5.3840e-58**	4.5314e-52
Trials reaching stopping criteria	**45/50**	32/50	42/50	38/50	50/50	50/50	50/50	50/50

Table 13: Experimental results for LDIW-PSO compared with DAPSO

Dim	Ackley			Griewank			Rastrigin		
	DAPSO	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$	*DAPSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$	*DAPSO*	*SSRDIW*$_{PSO}$	*SSRRIW*$_{PSO}$
20	3.906209e-014	**7.265299e-15**	3.291231e-02	8.605280e-002	3.096072e-02	**1.656290e-02**	2.159059e+001	1.994289e+01	**1.888908e+01**
30	4.159541e-008	1.323386e-14	**1.209699e-14**	2.583338e-002	**1.071397e-02**	1.303346e-02	3.263463e+001	2.996404e+01	**2.930789e+01**
40	7.046093e-005	2.907896e-14	**2.232881e-14**	1.087868e-002	**7.486326e-03**	9.839458e-03	3.890287e+001	**3.905068e+01**	3.978636e+01
50	1.025568e-003	2.482814e-13	**1.355893e-13**	1.346732e-002	**8.025957e-03**	1.169809e-02	4.823559e+001	4.426009e+01	**4.241095e+01**

Table 14a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 20

	Ackley			Griewank			Rastrigin		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW*$_{PSO}$	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW*$_{PSO}$	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW*$_{PSO}$
Best Fitness	7.5495e-15	3.9968e-15	3.9968e-15	1.9915e-02	7.3960e-03	1.2316e-02	2.9825e+00	2.9825e+00	2.9825e+00
Worst Fitness	1.3082e-11	7.5495e-15	7.5495e-15	3.7893e-01	2.1670e-01	1.6486e-01	3.3801e+01	3.8772e+01	2.3860e+01
Mean Fitness	9.2671e-13	4.7073e-15	**4.2810e-15**	1.1634e-01	8.5716e-02	**7.5072e-02**	1.1393e+01	1.1850e+01	**1.1145e+01**
Std. Dev.	2.1516e-12	1.4211e-15	**9.6383e-16**	6.6654e-02	4.2657e-02	**3.3104e-02**	5.3729e+00	5.3402e+00	**5.1549e+00**
SR (%)	100	100	100	11	22	24	100	100	100

Table 14b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 20

	Rosenbrock			**Schwefel 2.22**			**Sphere**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	1.9028e-01	5.8391e-01	5.0518e-02	2.3283e-16	7.3834e-32	4.9690e-38	3.2643e-27	2.1574e-56	9.8682e-69
Worst Fitness	8.8418e+00	9.8649e+00	8.3477e+01	5.5350e-13	4.1787e-24	1.4294e-32	3.6432e-22	2.7625e-47	6.0214e-60
Mean Fitness	4.4515e+00	**4.1277e+00**	4.3828e+00	3.3305e-14	4.2916e-26	**8.9304e-34**	1.7584e-23	4.9843e-49	**1.8318e-61**
Std. Dev.	**1.3834e+00**	1.5673e+00	8.1199+00	7.8644e-14	4.1571e-25	**2.3983e-33**	5.4979e-23	2.8213e-48	**8.2708e-61**
SR (%)	100	100	99	100	100	100	100	100	100

Table 15a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 30

	Ackley			**Griewank**			**Rastrigin**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	3.9968e-15	3.9968e-15	3.9968e-15	1.7226e-02	1.2316e-02	0.0000e+00	2.9825e+00	2.9825e+00	1.9883e+00
Worst Fitness	4.3388e-13	7.5495e-15	7.5495e-15	3.0732e-01	1.4269e-01	1.6232e-01	2.5848e+01	2.3860e+01	2.6842e+01
Mean Fitness	5.1568e-14	4.2810e-15	**4.2455e-15**	1.0222e-01	7.2816e-02	**6.9929e-02**	1.0608e+01	**9.6036e+00**	9.9019e+00
Std. Dev.	7.2065e-14	9.6383e-16	**9.0646e-16**	4.5279e-02	**2.9156e-02**	3.1923e-02	4.9350e+00	**4.1760e+00**	5.2057e+00
SR (%)	100	100	100	8	23	31	100	100	100

Table 15b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 30

	Rosenbrock			**Schwefel 2.22**			**Sphere**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	2.3881e-05	3.5978e-03	1.2181e-02	1.7600e-17	2.1097e-36	7.9370e-43	3.5127e-29	5.5836e-63	5.6464e-79
Worst Fitness	1.2759e+01	9.5548e+01	8.0678e+00	2.5894e-14	3.2170e-30	1.9570e-37	3.4672e-24	1.4025e52	1.4756e-68
Mean Fitness	4.4631e+00	6.3398e+00	**3.0391e+00**	1.6898e-15	5.9054e-32	**6.2312e-39**	7.5616e-26	1.6379e-54	**1.6617e-70**
Std. Dev.	1.7182e+00	1.4306e+01	**1.4428e+00**	3.8509e-15	3.2322e-31	**2.1735e-38**	3.5229e-25	1.4047e-53	**1.4716e-69**
SR (%)	100	97	100	100	100	100	100	100	100

Table 16a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 20

	Ackley			**Griewank**			**Rastrigin**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	2.0465E-10	7.5495e-15	7.5495e-15	0.0000e+00	0.0000e+00	0.0000e+00	1.0936e+01	1.3918e+01	1.8889e+01
Worst Fitness	4.6755E-08	1.3900e-13	1.7773e+00	6.6351e-02	5.1620e-02	7.1023e-02	8.0527e+01	6.7603e+01	6.7603e+01
Mean Fitness	2.7482E-09	**2.2364e-14**	1.7773e-02	1.4725e-02	1.3285e-02	**1.0621e-02**	3.2678e+01	3.3692e+01	**3.2489e+01**
Std. Dev.	5.2530E-09	**1.6979e-14**	1.7684e-01	1.6189e-02	1.4288e-02	**1.4009e-02**	1.0432e+01	9.5459e+00	**1.0170e+01**
SR (%)	100	100	99	94	97	97	95	96	92

Table 16b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 20

	Rosenbrock			**Schwefel 2.22**			**Sphere**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	1.6192e+01	2.0857e+00	5.7802e-01	5.2163e-13	2.2215e-23	4.3387e-28	1.0555e-19	1.5428e-38	1.1144e-53
Worst Fitness	2.1126e+02	8.5242e+01	8.0744e+01	6.1549e-09	3.2459e-15	1.6830e-17	5.9537e-15	1.8349e-31	1.5942e-43
Mean Fitness	3.3933e+01	2.7721e+01	**2.7449e+01**	1.5750e-10	3.9337e-17	**2.6515e-19**	2.8440e-16	3.7389e-33	**3.0056e-45**
Std. Dev.	2.5786e+01	**1.6612e+01**	1.9011e+01	6.8513e-10	3.2528e-16	**1.8366e-18**	7.8348e-16	2.3963e-32	**1.8304e-44**
SR (%)	86	91	87	100	100	100	100	100	100

Table 17a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 30

	Ackley			**Griewank**			**Rastrigin**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	3.5176e-12	7.5495e-15	7.5495e-15	0.0000e+00	0.0000e+00	0.0000e+00	1.3918e+01	1.2924e+01	1.1930e+01
Worst Fitness	6.3501e-10	2.8866e-14	2.1760e-14	8.2838e-02	8.5811e-02	5.1691e-02	5.5673e+01	5.4679e+01	5.0702e+01
Mean Fitness	1.3655e-10	1.4939e-14	**1.3589e-14**	1.5827e-02	1.4071e-02	**1.0187e-02**	**2.7926e+01**	2.8443e+01	2.8542e+01
Std. Dev.	1.3085e-10	4.1940e-15	**3.6750e-15**	1.7432e-02	1.6390e-02	**1.2222e-02**	**7.4278e+00**	8.2735e+00	9.1617e+00
SR (%)	100	100	100	94	96	99	99	97	98

Table 17b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 30

	Rosenbrock			**Schwefel 2.22**			**Sphere**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	1.2326e+01	1.6523e+00	7.3902e-01	7.5219e-15	9.4263e-29	7.0255e-37	1.3688e-22	1.7242e-45	3.1391e-64
Worst Fitness	1.1535e+02	8.6298e+01	1.4068e+02	5.0915e-12	2.6929e-23	2.7897-27	1.9235e-17	6.6331e-39	1.9224e-55
Mean Fitness	3.0978e+01	**2.7905e+01**	2.9223e+01	6.9822e-13	1.4820e-24	**3.7222e-29**	1.4619e-18	1.3751e-40	**3.3796e-57**
Std. Dev.	1.9755e+01	**1.7322e+01**	2.1778e+01	8.6910e-13	4.2054e-24	**2.8818e-28**	3.3268e-18	7.1578e-40	**2.0522e-56**
SR (%)	89	90	86	100	100	100	100	100	100

Table 18a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 20

	Ackley			**Griewank**			**Rastrigin**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	2.3631e-08	4.6629e-14	3.2419e-14	4.6185e-14	0.0000e+00	0.0000e+00	2.8831e+01	2.7837e+01	2.2866e+01
Worst Fitness	8.3629e-07	7.7741e-11	8.7875e-01	6.8700e-02	4.6483e-02	6.1166e-02	9.5440e+01	9.8422e+01	1.0439e+02
Mean Fitness	1.6804e-07	**1.4243e-12**	8.7875e-03	1.0084e-02	**7.7486e-03**	1.0543e-02	5.3377e+01	5.2412e+01	**5.2134e+01**
Std. Dev.	1.4161e-07	**7.9154e-12**	8.7434e-02	1.3541e-02	**1.0204e-02**	1.3527e-02	1.3992e+01	**1.3660e+01**	1.4172e+01
SR (%)	100	100	99	98	100	98	45	46	52

Table 18b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 20

	Rosenbrock			**Schwefel 2.22**			**Sphere**		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	2.9532e+01	2.6899e+01	2.1839e+01	1.0821e-10	6.6656e-18	7.6328e-22	6.1179e-15	4.3873e-29	5.4489e-39
Worst Fitness	1.5369e+02	1.6537e+02	1.4605e+02	1.4769e-05	7.0244e-11	5.6598e-11	5.3715e-11	2.6626e-21	2.3901e-31
Mean Fitness	**5.6306e+01**	5.7227e+01	6.1372e+01	1.9472e-07	8.6592e-13	**6.8063e-13**	4.3243e-12	2.7570e-23	**3.4195e-33**
Std. Dev.	**2.5463e+01**	2.7034e+01	2.6729e+01	1.4741e-06	7.0015e-12	**5.7339e-12**	8.2412e-12	2.6486e-22	**2.4105e-32**
SR (%)	79	71	61	100	100	100	100	100	100

Table 19a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 30

	Ackley			Griewank			Rastrigin		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	2.1934e-09	2.1760e-14	2.1760e-14	8.8818e-16	0.0000+00	0.0000e+00	2.2866e+01	2.1872e+01	2.3860e+01
Worst Fitness	1.0517e-07	1.5676e-13	9.2815e-14	3.9202e-02	4.1846e-02	5.8906e-02	8.8480e+01	9.9416e+01	7.0585e+01
Mean Fitness	1.8971e-08	3.8423e-14	**3.1992e-14**	**6.3007e-03**	7.4343e-03	7.4808e-03	4.7551e+01	**4.4717e+01**	4.4946e+01
Std. Dev.	1.9419e-08	1.9772e-14	**8.9070e-15**	**8.8591e-03**	1.0115e-02	1.1316e-02	1.2427e+01	1.1960e+01	**1.1342e+01**
SR (%)	100	100	100	100	100	99	55	72	70

Table 19b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 30

	Rosenbrock			Schwefel 2.22			Sphere		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	3.2034e+01	2.1319e+01	1.7313e+01	9.2580e-12	6.2036e-23	1.0544e-30	1.4142e-16	6.0959e-36	3.4990e-51
Worst Fitness	1.5805e+02	1.5195e+02	1.4811e+02	1.1639e-08	8.1066e-13	2.5765e-22	2.9783e-12	1.7368e-29	9.8013e-42
Mean Fitness	6.1470e+01	**5.3631e+01**	5.6316e+01	3.7523e-10	8.1180e-15	**6.8521e-24**	5.4458e-14	4.9748e-31	**1.0172e-43**
Std. Dev.	3.0565e+01	**2.4776e+01**	2.7229e+01	1.2318e-09	8.0659e-14	**3.2404e-23**	3.0355e-13	2.0485e-30	**9.7490e-43**
SR (%)	72	78	71	100	100	100	100	100	100

Table 20: Performance ranking in terms of mean best fitness for LDIW-PSO, CDIW-PSO and $SSRDIW_{PSO}$

Test Problem	Swarm Size	Dimension = 10			Dimension = 30			Dimension = 50		
		LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Ackley	20	3	2	1	2	1	3	2	1	3
	30	3	2	1	3	2	1	3	2	1
Griewank	20	3	2	1	3	2	1	2	1	3
	30	3	2	1	3	2	1	1	2	3
Rastrigin	20	2	3	1	2	3	1	3	2	1
	30	3	1	2	1	2	3	3	1	2
Rosenbrock	20	3	1	2	3	2	1	1	2	3
	30	2	3	1	3	1	2	3	1	2
Schwefel P2.22	20	3	2	1	3	2	1	3	2	1
	30	3	2	1	3	2	1	3	2	1
sphere	20	3	2	1	3	2	1	3	2	1
	30	3	2	1	3	2	1	3	2	1
Average		5.67	4.00	**2.33**	5.33	3.83	**2.83**	5.00	**3.33**	3.67

Table 21a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 20

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	2.6682e-04	3.9968e-15	3.9968e-15	5.7696e-02	9.8573e-03	1.2321e-02	2.9826e+00	3.9767e+00	2.9825e+00
Worst Fitness	3.6701e-02	1.6456e+00	7.5495e-15	5.2743e-01	2.3098e-01	2.1171e-01	2.8831e+01	3.0951e+01	2.3860e+01
Mean Fitness	6.3572e-03	1.6456e-02	**4.4587e-15**	2.65073-01	**8.0140e-02**	8.1509e-02	1.1970e+01	1.1792e+01	**1.0518e+01**
Std. Dev.	6.1370e-03	1.6374e-01	**1.1948e-15**	1.2088e-01	4.3710e-02	**4.1818e-02**	5.4829e+00	5.2183e+00	**4.4331e+00**
SR (%)	9	99	100	0	23	21	100	100	100

Table 21b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 20

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	5.0327e-01	2.3924e-06	1.3474e-01	7.5130e-05	2.0291e-24	2.2651e-33	5.2270e-06	2.0572e-42	3.9833e-60
Worst Fitness	1.6783e+02	9.2790e+00	1.0544e+02	1.7082e-02	2.1966e-19	1.3014e-28	2.2563e-03	1.5891e-31	4.1936e-51
Mean Fitness	1.1295e+01	**4.1748e+00**	4.4782e+00	2.0124e-03	9.2187e-21	**4.8514e-30**	3.5628e-04	3.6683e-33	**6.3339e-53**
Std. Dev.	2.3681e+01	1.9017e+00	**1.0294e+01**	2.3871e-03	3.1553e-20	**1.6657e-29**	4.8987e-04	1.8265e-32	**4.3016e-52**
SR (%)	96	100	99	38	100	100	88	100	100

Table 22a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 30

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	5.0628e-04	3.9968e-15	3.9968e-15	5.3134e-02	9.8573e-03	7.3960e-03	2.9825e+00	1.9883e+00	2.9825e+00
Worst Fitness	1.9113e-02	7.5495e-15	7.5495e-15	5.4171e-01	1.7455e-01	1.6488e-01	2.6843e+01	2.7837e+01	1.9883e+01
Mean Fitness	4.5976e-03	4.2100e-15	**4.1034e-15**	2.2761e-01	7.0440e-02	**6.8499e-02**	1.0946e+01	**9.7826e+00**	9.8223e+00
Std. Dev.	3.9745e-03	8.4372e-16	**6.0605e-16**	1.1050e-01	3.4206e-02	**3.3744e-02**	4.6641e+00	5.2431e+00	**3.7099e+00**
SR (%)	4	100	100	0	34	34	100	100	100

Table 22b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 10 dimensions with swarm size of 30

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	2.7771e-01	4.2983e-03	2.1897e-02	4.4332e-05	5.9217e-27	7.7888e-37	1.7477e-06	1.3646e-45	4.9571e-67
Worst Fitness	1.0266e+02	9.3917e+01	8.1281e+00	5.1963e-03	7.6797e-22	6.9684e-33	1.2363e-03	1.2652e-34	2.9736e-58
Mean Fitness	7.3222e+00	4.6473e+00	**3.0159e+00**	1.1568e-03	5.2086e-23	**2.7837e-34**	1.7524e-04	1.4248e-36	**3.5758e-60**
Std. Dev.	9.7155e+00	9.1254e+00	**1.8573e+00**	9.8438e-04	1.2312e-22	**7.6303e-34**	2.4192e-04	1.2594e-35	**2.9790e-59**
SR (%)	99	99	100	57	100	100	97	100	100

Table 23a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 20

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	7.6075e-02	1.0347e-13	7.5495e-15	3.7072e-01	0.0000e+00	0.0000e+00	1.3497e+01	1.2924e+01	1.0936e+01
Worst Fitness	5.8770e-01	5.4655e-11	1.6456e+00	1.0221e+00	1.1059e-01	7.8665e-02	7.6612e+01	5.7661e+01	5.8656e+01
Mean Fitness	2.8934e-01	**7.0854e-12**	6.2645e-02	8.3199e-01	1.5365e-02	**1.3793e-02**	3.6975e+01	3.1803e+01	**3.1068e+01**
Std. Dev.	1.1061e-01	**1.1089e-11**	2.7657e-01	1.3791e-01	1.9516e-02	**1.5409e-02**	1.2344e+01	**9.5081e+00**	1.0160e+01
SR (%)	0	100	95	0	92	98	82	96	94

Table 23b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 20

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	3.1445e+01	5.3172e+00	3.4264e+00	9.4209e-02	3.6066e-19	4.2575e-29	1.4671e-01	3.1570e-25	4.9847e-48
Worst Fitness	3.8305e+02	1.0504e+02	8.5352e+01	4.8363e-01	3.7281e-14	5.1517e-23	3.5501e+00	1.5072e-20	1.2989e-40
Mean Fitness	9.0321e+01	3.3052e+01	**3.0608e+01**	2.6077e-01	1.3696e-15	**2.2417e-24**	9.9836e-01	9.3610e-22	**1.6515e-42**
Std. Dev.	6.5995e+01	2.2626e+01	**2.1631e+01**	8.4975e-02	4.6744e-15	**7.1414e-24**	5.1607e-01	2.3631e-21	**1.2937e-41**
SR (%)	34	82	81	0	100	100	0	100	100

Table 24a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 30

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	1.0369e-01	1.4655e-14	7.5495e-15	2.5742e-01	0.0000e+00	0.0000e+00	1.1137e+01	1.4912e+01	8.9475e+00
Worst Fitness	5.8677e-01	1.4286e-12	2.1760e-14	1.0162e+00	6.8642e-02	6.1273e-02	5.6933e+01	5.8656e+01	6.5615e+01
Mean Fitness	2.3662e-01	1.1100e-13	**1.2026e-14**	8.0305e-01	1.2720e-02	**1.2227e-02**	3.0321e+01	**2.8910e+01**	2.9656e+01
Std. Dev.	7.2953e-02	1.9207e-13	**3.7806e-15**	1.3412e-01	1.3742e-02	**1.3654e-02**	8.9028e+00	**8.8817e+00**	9.7347e+00
SR (%)	0	100	100	0	96	96	97	98	96

Table 24b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 30 dimensions with swarm size of 30

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*$SSRRIW_{PSO}$*
Best Fitness	3.0042e+01	7.5603e+00	6.5598e+00	9.4323e-02	1.0508e-21	9.3322e-35	1.7171e-01	7.1988e-31	4.6258e-58
Worst Fitness	3.5254e+02	1.0744e+02	8.5689e+01	4.5703e-01	6.2399e-17	1.8446e-29	3.2880e+00	4.2228e-22	1.2856e-50
Mean Fitness	8.0443e+01	3.4158e+01	**2.9701e+01**	2.0920e-01	2.0142e-18	**3.3125e-31**	9.0064e-01	5.2226e-24	**3.1736e-52**
Std. Dev.	6.1967e+01	2.3257e+01	**2.0574e+01**	7.4517e-02	6.9944e-18	**1.9947e-30**	4.7463e-01	4.2099e-23	**1.6311e-51**
SR (%)	54	79	84	0	100	100	0	100	100

Table 25a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 20

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	3.7363e-01	9.1547e-11	2.1760e-14	9.9927e-01	0.0000e+00	0.0000e+00	3.2130e+01	3.0819e+01	2.4854e+01
Worst Fitness	1.2499e+00	2.5416e-07	2.0126e+00	1.1542e+00	5.1442e-02	8.5172e-02	9.3415e+01	1.0638e+02	8.9475e+01
Mean Fitness	7.6250e-01	**1.1668e-08**	1.0014e-01	1.0770e+00	**8.6034e-03**	1.3725e-02	5.9013e+01	5.1358e+01	**4.9748e+01**
Std. Dev.	2.0357e-01	3.0443e-08	4.0343e-01	2.4943e-02	1.2163e-02	1.7229e-02	1.2692e+01	1.3579e+01	1.4310e+01
SR (%)	0	100	94	0	99	97	22	50	54

Table 25b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 20

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	1.4630e+02	2.9319e+01	3.9227e+00	6.0443e-01	1.4917e-14	4.9312e-23	3.9011e+00	1.3119e-18	1.3133e-38
Worst Fitness	1.3202e+03	1.5698e+02	1.5186e+02	1.5538e+00	1.8198e-10	8.6419e-16	1.7889e+01	4.9716e-14	8.3601e-31
Mean Fitness	3.1463e+02	7.2915e+01	**5.9867e+01**	1.0389e+00	3.5025e-12	**9.8776e-18**	8.9417e+00	2.9163e-15	**9.9652e-33**
Std. Dev.	1.6544e+02	3.0272e+01	3.0551e+01	2.3005e-01	1.8487e-11	8.6379e-17	3.0265e+00	7.8862e-15	8.3149e-32
SR (%)	0	47	66	0	100	100	0	100	100

Table 26a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 30

	Ackley			Griewank			Rastrigin		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	3.0381e-01	1.0183e-11	1.4655e-14	7.7108e-01	0.0000e+00	0.0000e+00	2.2200e+01	1.8889e+01	2.5848e+01
Worst Fitness	1.1737e+00	8.6233e-09	1.5607e+00	1.1282e+00	6.3542e-02	1.1634e-01	8.1841e+01	8.7486e+01	8.3510e+01
Mean Fitness	6.6528e-01	**5.6290e-10**	2.5874e-02	1.0586e+00	**7.3540e-03**	1.1541e-02	4.8061e+01	**4.3902e+01**	4.3972e+01
Std. Dev.	1.7192e-01	1.2606e-09	1.8501e-01	3.9840e-02	1.0921e02	1.7802-02	1.1406e+01	1.2159e+01	1.0683e+01
SR (%)	0	100	98	0	99	96	63	77	77

Table 26b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the scalable benchmark problems in 50 dimensions with swarm size of 30

	Rosenbrock			Schwefel 2.22			Sphere		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	1.1525e+02	1.0825e+01	1.4610e+01	5.1310e-01	7.6172e-17	4.5557e-30	2.8877e+00	3.6199e-21	9.2338e-48
Worst Fitness	8.9680e+02	1.4977e+02	1.4609e+02	1.5876e+00	2.0753e-13	3.3717e-25	1.6115e+01	2.1283e-16	4.1533e-41
Mean Fitness	2.7098e+02	6.7473e+01	**6.0529e+01**	9.3819e-01	6.8685e-15	**2.2273e-26**	7.2065e+00	7.7623e-18	**6.4333e-43**
Std. Dev.	1.1521e+02	3.1220e+01	**2.9462e+01**	2.1819e-01	2.2435e-14	**5.3956e-26**	2.4652e+00	2.3964e-17	**4.1989e-42**
SR (%)	0	58	61	0	100	100	0	100	100

Table 27: Performance ranking in terms of mean best fitness for RIW-PSO, CRIW-PSO and $SSRRIW_{PSO}$

Test Problem	Swarm Size	Dimension = 10			Dimension = 30			Dimension = 50		
		RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Ackley	20	2	3	1	3	1	2	3	1	2
	30	3	2	1	3	2	1	3	1	2
Griewank	20	3	1	2	3	2	1	3	1	2
	30	3	2	1	3	2	1	3	1	2
Rastrigin	20	3	2	1	3	2	1	3	2	1
	30	3	1	2	3	1	2	3	1	2
Rosenbrock	20	3	1	2	3	2	1	3	2	1
	30	3	2	1	3	2	1	3	2	1
Schwefel P2.22	20	3	2	1	3	2	1	3	2	1
	30	3	2	1	3	2	1	3	2	1
sphere	20	3	2	1	3	2	1	3	2	1
	30	3	2	1	3	2	1	3	2	1
Average		5.83	3.67	**2.50**	6.00	3.67	**2.33**	6.00	3.17	**2.83**

Table 28a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the low-scaled benchmark problems with swarm size of 20

	Booth			Esom			Michalewicz 5		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-4.0236e+00	-36547e+00	-3.6419e+00
Worst Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-1.6676e+00	-1.7707e+00	-1.6865e+00
Mean Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-2.5887e+00	-2.6595e+00	**-2.7246e+00**
Std. Dev.	0.0000e+00	0.0000e+00	0.0000e+00	1.4433e-15	1.4433e-15	1.4433e-15	4.2216e-01	3.8361e+00	3.8299e-01
SR (%)	100	100	100	100	100	100	0	0	0

Table 28b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the low-scaled benchmark problems with swarm size of 20

	Schaffer's f6			Shubert			Trid 6		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.8673e+02	-18673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Worst Fitness	9.7159e-03	9.7159e-03	9.7159e-03	-1.8673e+02	-1.2358e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Mean Fitness	**1.9432e-04**	3.1091e-03	4.4943e-03	-1.8673e+02	-1.8610e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Std. Dev.	1.3602e-03	4.5322e-03	4.8257e-03	1.9479e-13	6.2838e+00	1.3999e-13	7.2196e-14	7.0153e-14	8.2042e-14
SR (%)	98	68	53	100	99	100	100	100	100

Table 29a: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the low-scaled benchmark problems with swarm size of 30

	Booth			Esom			Michalewicz 5		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-3.7093e+00	-3.9666e+00	-4.0414e+00
Worst Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-1.9647e+00	-2.0913e+00	-1.8581e+00
Mean Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-2.7524e+00	-2.9189e+00	**-2.9281e+00**
Std. Dev.	0.0000e+00	0.0000e+00	0.0000e+00	1.4433e-15	1.4433e-15	1.4433e-15	3.3793e-01	3.5757e-01	3.8822e-01
SR (%)	100	100	100	100	100	100	0	0	0

Table 29b: Results for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO for the low-scaled benchmark problems with swarm size of 30

	Schaffer's f6			Shubert			Trid 6		
	LDIW-PSO	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*	*LDIW-PSO*	*CDIW-PSO*	*SSRDIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Worst Fitness	0.0000e+00	9.7159e-03	9.7159e-03	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Mean Fitness	**0.0000e+00**	2.0403e-03	3.5171e-03	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Std. Dev.	0.0000e+00	3.9574e-03	4.6530e-03	1.1548e-13	1.9915e-13	1.1132e-13	6.7273e-14	8.1796e-14	7.4455e-14
SR (%)	100	79	63	100	100	100	100	100	100

Table 30a: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the low-scaled benchmark problems with swarm size of 20

	Booth			Esom			Michalewicz 5		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-3.8987e+00	-36182e+00	-4.1505e+00
Worst Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-1.9726e+00	-2.0289e+00	2,0063e+00
Mean Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-2.7105e+00	-2.7591e+00	**-2.8714e+00**
Std. Dev.	0.0000e+00	0.0000e+00	0.0000e+00	1.4479e-15	1.4433e-15	1.4433e-15	3.6438e-01	3.6934e-01	4.0069e-01
SR (%)	100	100	100	100	100	100	0	0	0

Table 30b: Results for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO for the low-scaled benchmark problems with swarm size of 20

	Schaffer's f6			Shubert			Trid 6		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.8673e+02	-1.8673e+02	-18673e+02	-5.0000E+01	-5.0000e+01	-5.0000e+01
Worst Fitness	9.7159e-03	9.7159e-03	9.7159e-03	-1.8673e+02	-1.8673e+02	-1.2358e+02	-5.0000E+01	-5.0000e+01	-5.0000e+01
Mean Fitness	**8.7533e-04**	4.8580e-03	5.4414e-03	**-1.8673e+02**	**-1.8673e+02**	-1.8610e+02	-5.0000E+01	-5.0000e+01	-5.0000e+01
Std. Dev.	2.7802e-03	4.8580e-03	4.8223e-03	2.2081e-13	8.4933e-14	6.2838e+00	2.2079e-07	7.4455e-14	**7.1747e-14**
SR (%)	90	50	43	100	100	99	100	100	100

Table 31a: Results for SSRRIW$_{PSO}$, CRIW-PSO and RIW-PSO for the low-scaled benchmark problems with swarm size of 30

	Booth			**Esom**			**Michalewicz 5**		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-3.9345e+00	-4.4362e+00	-3.9058e+00
Worst Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-2.0385e+00	-2.1389e+00	-2.4530e+00
Mean Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.0000e+00	-1.0000e+00	-1.0000e+00	-2.8436e+00	-2.9961e+00	**-3.0545e+00**
Std. Dev.	0.0000e+00	0.0000e+00	0.0000e+00	1.4444e-15	1.4433e-15	1.4433e-15	3.4103e-01	3.7627e-01	3.4268e-01
SR (%)	100	100	100	100	100	100	0	0	0

Table 31b: Results for SSRRIW$_{PSO}$, CRIW-PSO and RIW-PSO for the low-scaled benchmark problems with swarm size of 30

	Schaffer's f6			**Shubert**			**Trid 6**		
	RIW-PSO	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*	*RIW-PSO*	*CRIW-PSO*	*SSRRIW$_{PSO}$*
Best Fitness	0.0000e+00	0.0000e+00	0.0000e+00	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Worst Fitness	9.7159e-03	9.7159e-03	9.7159e-03	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Mean Fitness	**9.7159e-05**	3.1091e-03	4.5674e-03	-1.8673e+02	-1.8673e+02	-1.8673e+02	-5.0000e+01	-5.0000e+01	-5.0000e+01
Std. Dev.	9.6672e-04	4.5322e-03	4.8484e-03	1.3391e-13	1.9924e-13	9,1525e-14	4.2546e-08	6.3553e-14	7.3458e-14
SR (%)	99	68	52	100	100	100	100	100	100

```
Begin PSO Algorithm
   Input:  f: the function to optimize
           s: the swarm size
           n: the problem dimension
           pr: solution search space
           vr: particle velocity range
   Output: x*: the best particle position found
           f*: the best fitness value found
   Initialize: position x_i = (x_i1, …, x_in) and velocity v_i = (v_i1, …, v_in), for all
   particles in problem space
   evaluate f(x_i) in n variables and get pbest_i, (i = 1, …, s)
   gbest ← best of pbest_i
   While stopping criteria is false do
      succ ← 0
      Loop for s times
       Loop for n times
           calculate ω using equation (8) or (9)
           update v_i for particle using equation (1)
           validate for velocity boundaries based on vr
           update x_i for particle using equation (2)
           validate for position boundaries based on pr
        End
      End
      If f(x_i) < f(pbest_i) then
           pbest_i ← x_i
           succ ← succ + 1
      end if
      If f(x_i) < f(gbest) then
           gbest ← x_i
           f(gbest) ← f(x_i)
      end if
      compute swarm success rate using equation (4)
   End while
   x* ← gbest
   f* ← f(gbest)
   Return x* and f*
End PSO Algorithm
```

Figure 1: Inertia weight PSO algorithm for $SSRDIW_{PSO}$ and $SSRRIW_{PSO}$

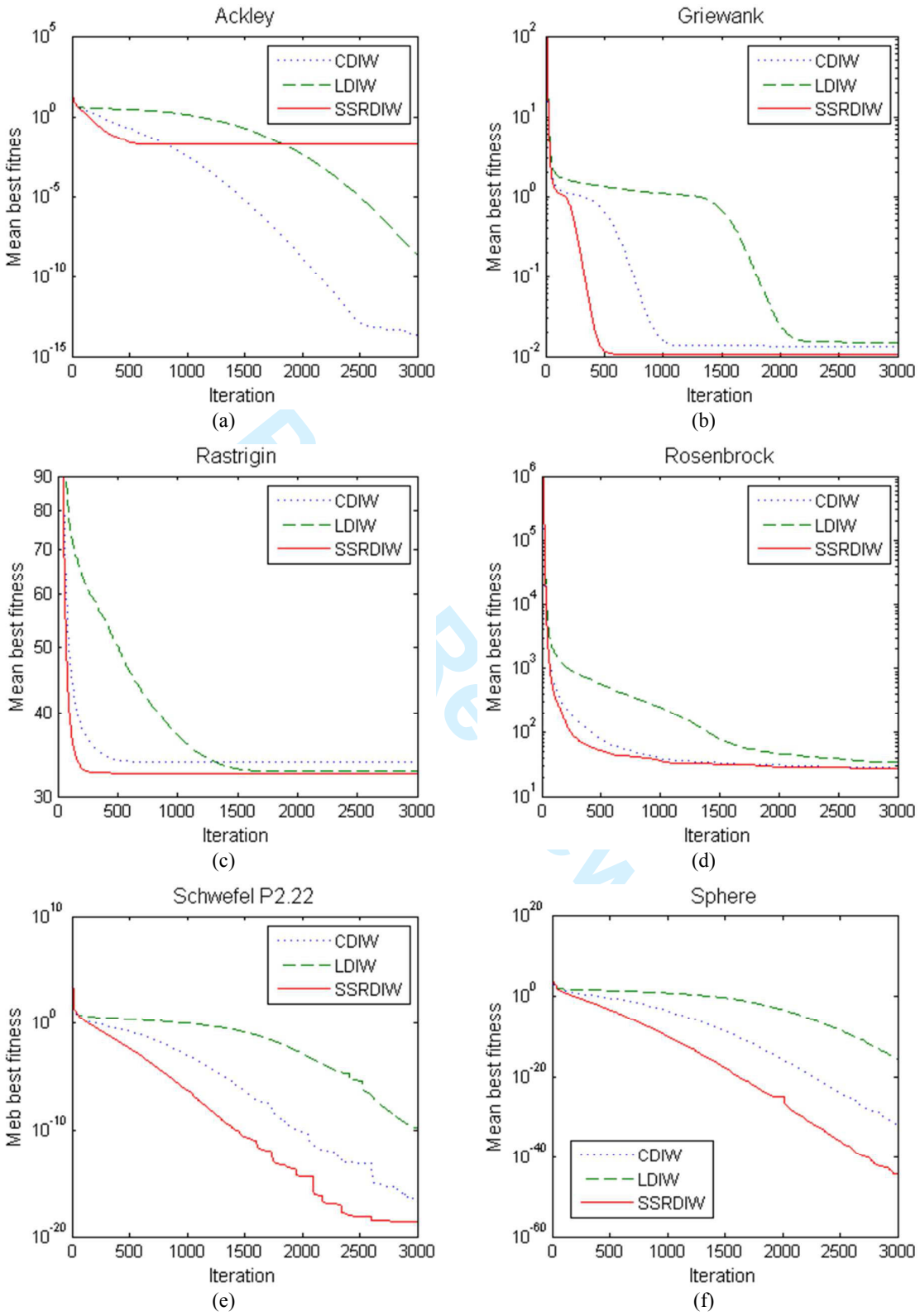


Figure 2: Convergence curves for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO in six test problems with *size = 20* and *dim = 30*

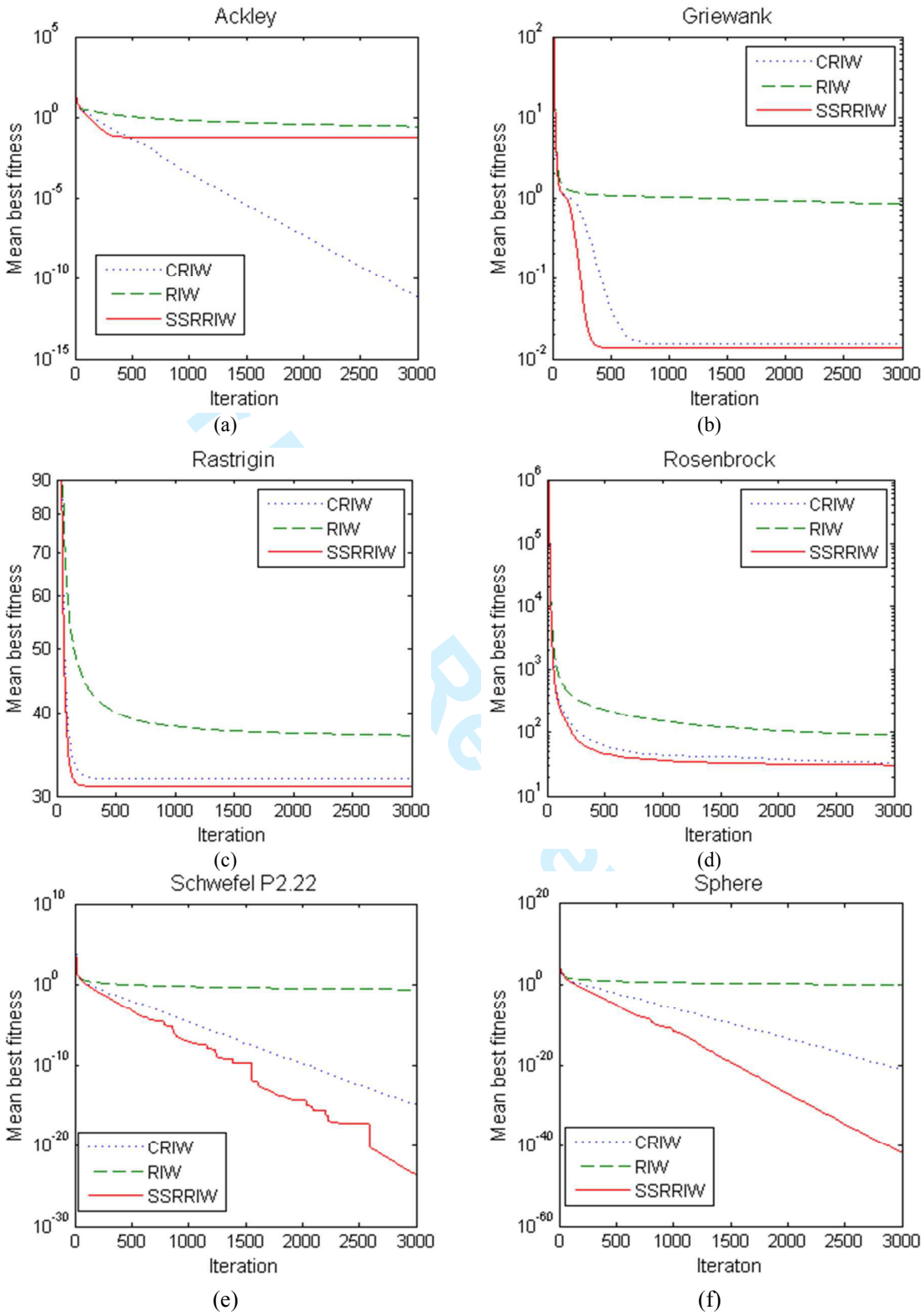


Figure 3: Convergence curves for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO in six test problems with *size = 20* and *dim = 30*

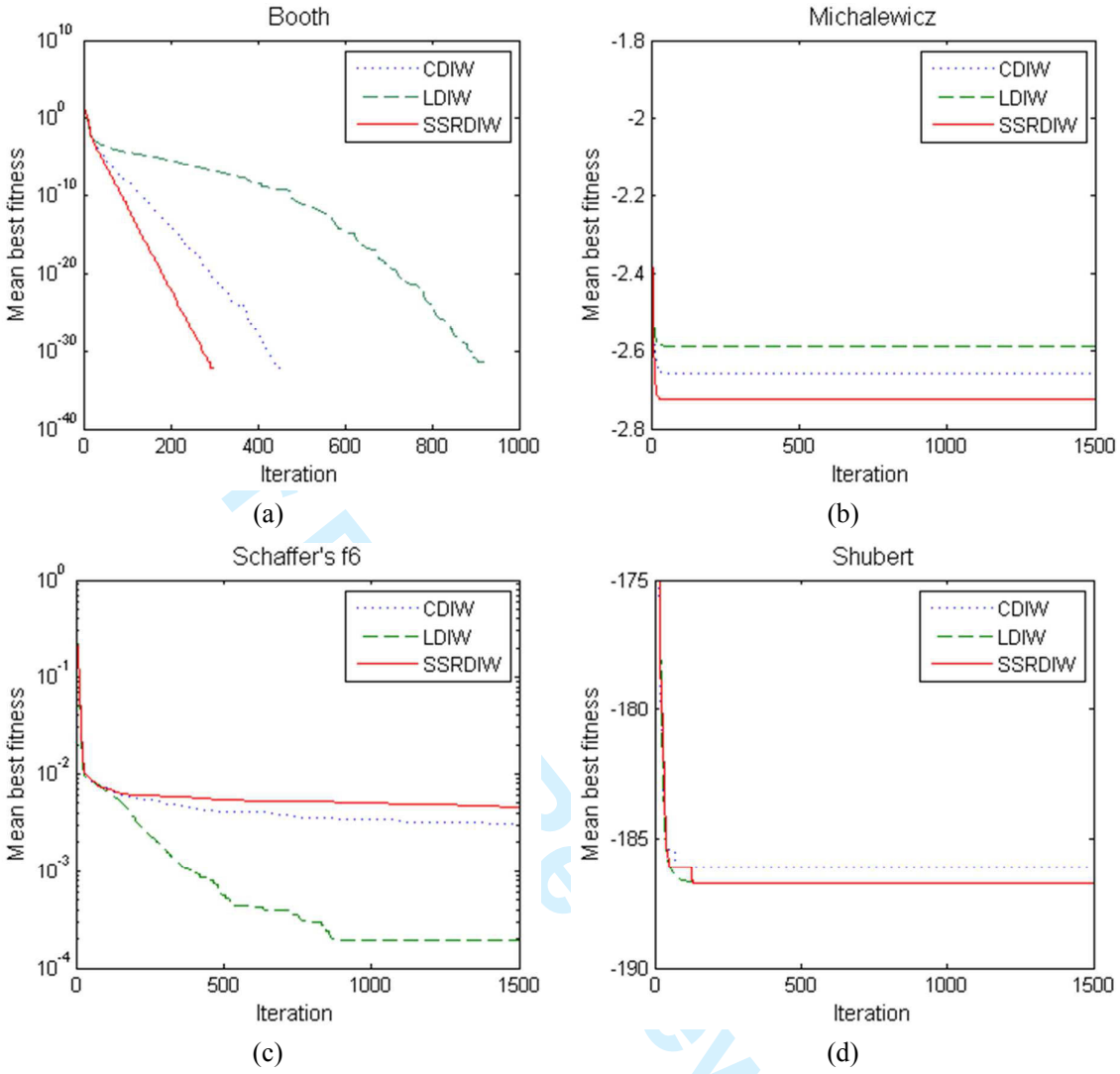


Figure 4: Convergence curves for $SSRDIW_{PSO}$, CDIW-PSO and LDIW-PSO in four low-scaled test problems with *size = 20*

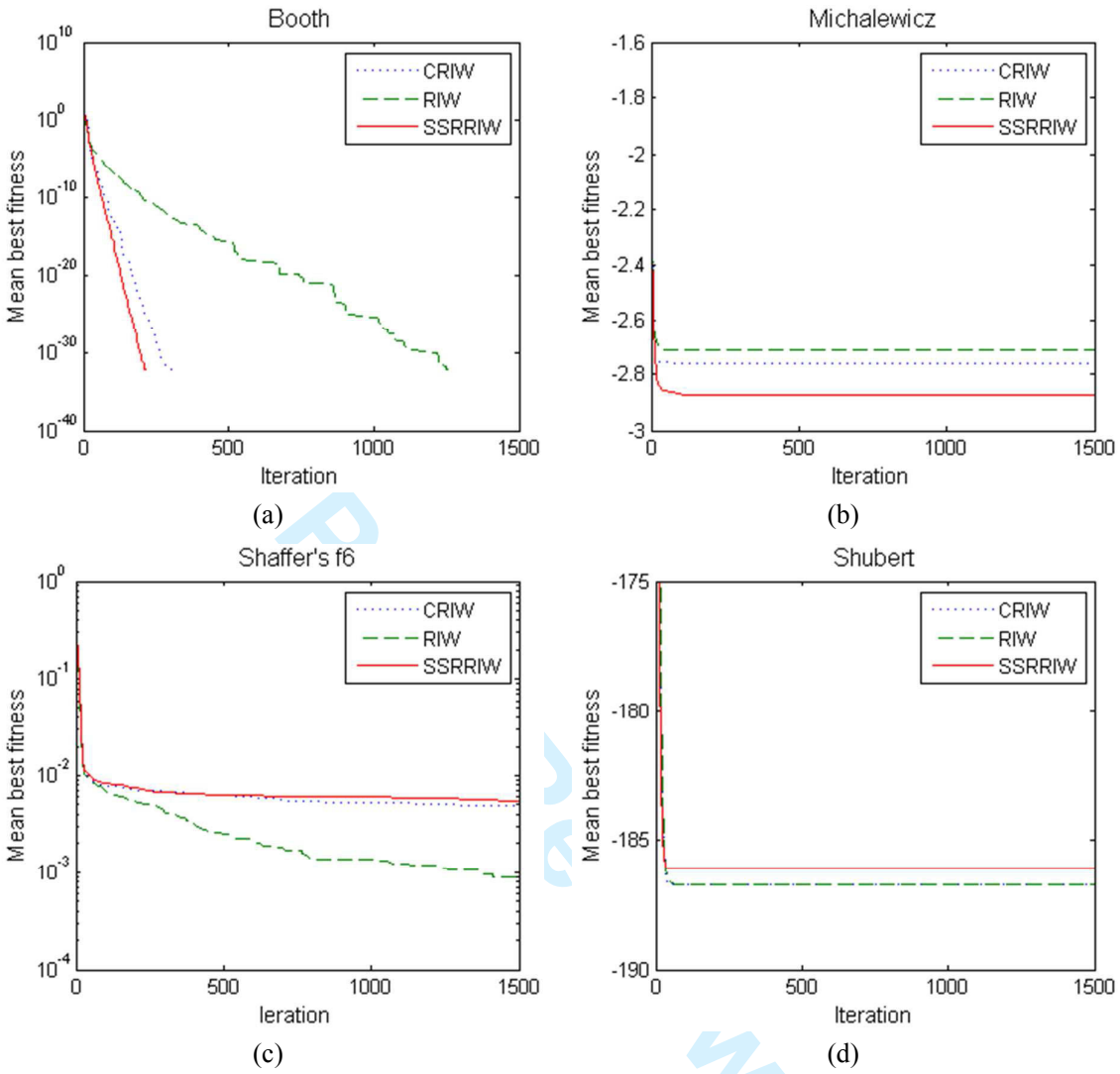


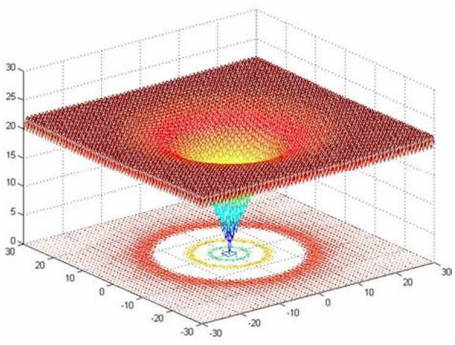
Figure 5: Convergence curves for $SSRRIW_{PSO}$, CRIW-PSO and RIW-PSO in four low-scaled test problems with *size = 20*

Appendix I: *Descriptions of the Benchmark Global Optimization Test Problems used in the experiments*

Described below are the benchmark global optimization problems used in the experiments. The mathematical models as well as graphical representations of these problems are also given. The essence of the graphs is to facilitate the comprehension of the landscape of the respective problem. The test problems are grouped into two – *High-scaled* and *Low-scaled*. Additional information about the problems was obtained from *https://en.wikipedia.org/wiki/Test_functions_for_optimization* and *http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm*.

1. High-scaled global optimization test problems

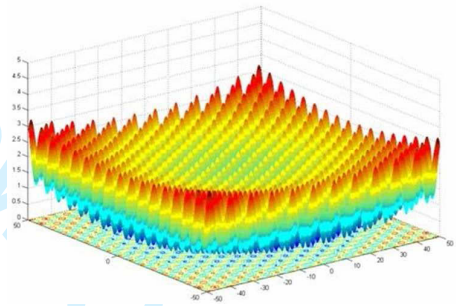
Ackley problem: It is continuous, multimodal, scalable and non-separable. It is a widely used test problem. The global minimum $f_1(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$ and the number of local minima is not known. Search space is in [-30,30].



Model:

$$f(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$$

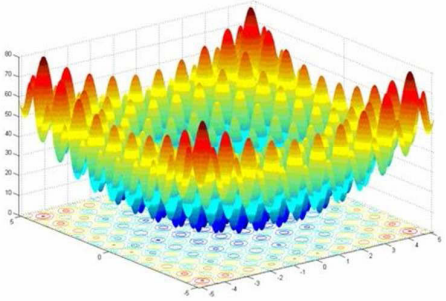
Griewank problem: Similar to Rastrigin. It is continuous, multimodal, scalable and non-separable with many widespread local minima regularly distributed. The complexity of the problem increases with its dimensionality. Its global minimum $f_2(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$ and the number of local minima for arbitrary n is not known, but in the two dimensional case, there are some 500 local minima. Search space is in [-600,600].



Model:

$$f(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$$

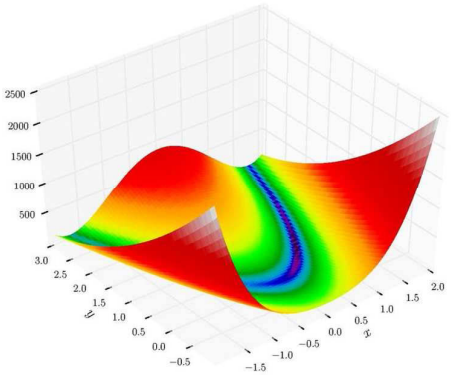
Rastrigin problem: It is continuous, multimodal, scalable and separable with many local minima arranged in a lattice-like configuration. It is based on the Sphere problem with the addition of cosine modulation so as to produce frequent local minima. There are about 50 local minima for two dimensional case and its global minimum $f_3(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$. Search space is in [-5.12,5.12].



Model:

$$f(\vec{x}) = \sum_{i=1}^{d}(x_i^2 - 10\cos(2\pi x_i) + 10)$$

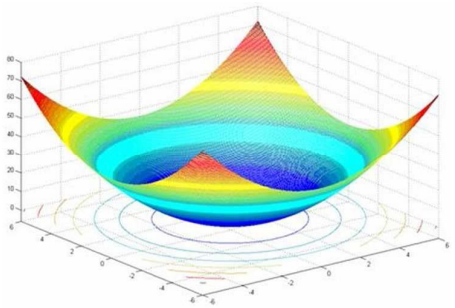
Rosenbrock problem: It is continuous, unimodal, scalable and non-separable. It is a classic optimization problem also known as banana function, the second function of De Jong or extended Rosenbrock function. Its global minimum $f_4(\vec{x}) = 0$ obtainable at $\vec{x} = 1$, lies inside a long narrow, parabolic shaped valley. Though it look simple to solve, yet due to a saddle point it is very difficult to converge to the global optimum. Search space is in [-30,30].



Model:

$$f(\vec{x}) = \sum_{i=1}^{d-1} (100(x_{i+1} - x_i^2)^2) + (x_i - 1)^2$$

Sphere problem: Known as the first De Jong function is continuous, convex, unimodal, scalable and separable. It is one of the simplest test benchmark problems. Its global minimum $f_6(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$. Search space is in [-100,100].

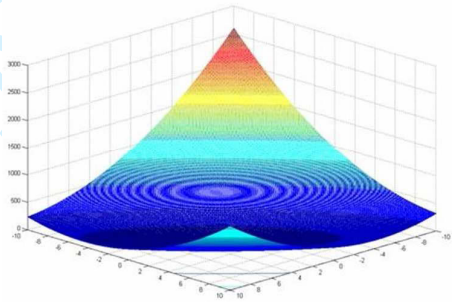


Model:

$$f(\vec{x}) = \sum_{i=1}^{d} x_i^2$$

2. Low-scaled global optimization test problems

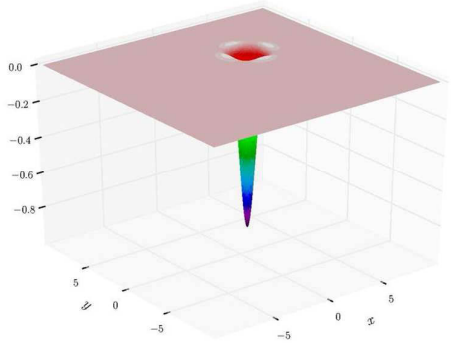
Booth problem: It is continuous, multimodal, non-scalable and separable. Its global minimum $f_5(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$. Search space is in [-10,10].



Model:

$$f(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

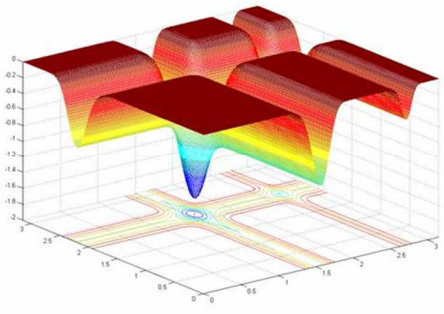
Easom problem: It is continuous, unimodal, non-scalable and non-separable. Its global minimum has a small area relative to the search space. It was inverted for minimization and has only two variables. Its global minimum $f_5(\vec{x}) = -1$ is obtainable at $\vec{x} = \pi$. Search space is in [-100,100].



Model:

$$f(\vec{x}) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

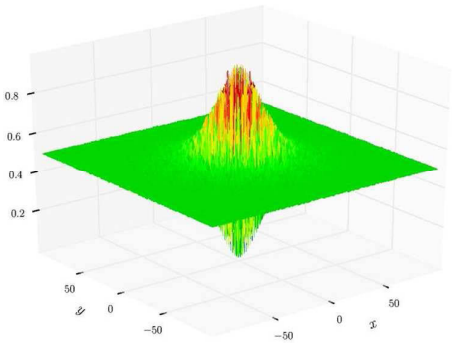
Michalewicz problem: It is continuous, multimodal and separable. It has *d*! local optima. The parameter *m* defines the "steepness" of the valleys or edges. Larger *m* leads to more difficult search. For very large *m* the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum. The value of *m* is usually 10. The approximated global minimum is $f_5(\vec{x}) = -4.687$ for *d* = 5 and $f_5(\vec{x}) = -9.66$ for *d* = 10. Search space is in [0,π]



Model:

$$f(\vec{x}) = -\sum_{i=1}^{d} \sin(x_i)\left[sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$$

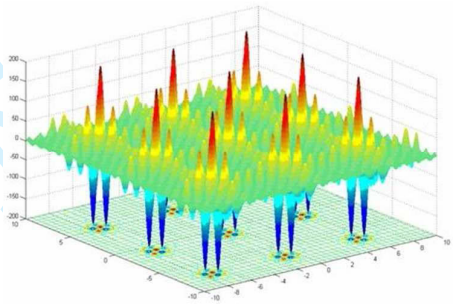
Schaffer's f6 problem: It is 2-dimensional, continuous, multimodal and non-separable with unknown number of many local minima. Its global minimum $f_5(\vec{x}) = 0$ is obtainable at $\vec{x}$ = 0. Search pace is in [-100,100].



Model:

$$f(\vec{x}) = \sum_{i=1}^{d-1}\left(0.5 + \frac{sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2}\right)$$

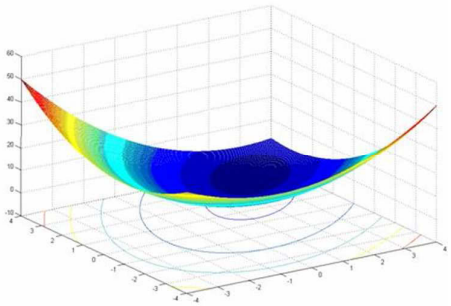
Shubert's problem: It is 2-dimensional, continuous, multimodal and non-separable with unknown number of local minima. But with *d* = 2, there are 760 local optimal, 18 of which are global with $f_5(\vec{x}) = -1.86.7309$. Search pace is in [-10,10].



Model:

$$f(\vec{x}) = \prod_{i=1}^{d}\left(\sum_{j=1}^{5} jcos((j+1)x_i + j)\right)$$

Trid problem: It is continuous, unimodal and non-separable. It has no local optima except the global one. With *d* = 6, the global minimum $f_5(\vec{x}) = -50$ but with *d* = 10, the global minimum $f_5(\vec{x}) = -200$. Search pace is in [$-d^2$,d^2].



Model:

$$f(\vec{x}) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d} x_i x_{i-1}$$

Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 178959, 17 pages
http://dx.doi.org/10.1155/2014/178959

Research Article

An Investigation into the Performance of Particle Swarm Optimization with Various Chaotic Maps

Akugbe Martins Arasomwan and Aderemi Oluyinka Adewumi

School of Mathematics, Statistics & Computer Science, University of KwaZulu-Natal, Private Bag X54001, Durban 4000, South Africa

Correspondence should be addressed to Aderemi Oluyinka Adewumi; laremtj@gmail.com

Received 12 September 2013; Accepted 9 December 2013; Published 20 January 2014

Academic Editor: Sergio Preidikman

This paper experimentally investigates the effect of nine chaotic maps on the performance of two Particle Swarm Optimization (PSO) variants, namely, Random Inertia Weight PSO (RIW-PSO) and Linear Decreasing Inertia Weight PSO (LDIW-PSO) algorithms. The applications of logistic chaotic map by researchers to these variants have led to Chaotic Random Inertia Weight PSO (CRIW-PSO) and Chaotic Linear Decreasing Inertia Weight PSO (CDIW-PSO) with improved optimizing capability due to better global search mobility. However, there are many other chaotic maps in literature which could perhaps enhance the performances of RIW-PSO and LDIW-PSO more than logistic map. Some benchmark mathematical problems well-studied in literature were used to verify the performances of RIW-PSO and LDIW-PSO variants using the nine chaotic maps in comparison with logistic chaotic map. Results show that the performances of these two variants were improved more by many of the chaotic maps than by logistic map in many of the test problems. The best performance, in terms of function evaluations, was obtained by the two variants using Intermittency chaotic map. Results in this paper provide a platform for informative decision making when selecting chaotic maps to be used in the inertia weight formula of LDIW-PSO and RIW-PSO.

1. Introduction

PSO algorithm is one of the many algorithms that have been proposed over the years for global optimization. When it was proposed in 1995 [1], swarm size, particle velocity, acceleration coefficients, and random coefficients were the associated parameters that controlled its operations. A close look at the algorithm shows that randomness plays very useful role in making the algorithm effectively solve optimization problems. Randomness comes into play at the point of initializing the particles in the solution space and in updating the velocities of particles at each iteration of the algorithm. This random feature has contributed immensely to the performance of PSO [1–3]. To further enhance the performance of PSO, inertia weight strategy (IWS) was introduced into it by [4] to facilitate the intensification and diversification characteristics of the algorithm. Intensification searches around the current best solutions and selects the best candidate, while diversification makes the algorithm explore the search space more efficiently, mostly by means of randomization. As a result, randomness has been brought into the IWS by different researchers [5–8]. The important role of randomization can also be played by using chaos theory. Chaos is mathematically defined as randomness generated by simple deterministic system [2]. It is generally characterised by three dynamic properties, namely, ergodicity, stochastic, and sensitivity, to its initial conditions [2, 9]. These characteristics can enhance the search ability of PSO. This seems to be the motivation behind the introduction of chaos feature into IWS in [10], which led to improved optimizing capabilities of CDIW-PSO and CRIW-PSO due to better global search mobility compared with LDIW-PSO and RIW-PSO, respectively. Chaos optimizations have been applied to different aspects of PSO by various researchers over the years [9, 11–14]. In order to increase the diversity of the swarm and prevent premature convergence to local optimal, chaos mutation operator based on logistic map was used in [13] and another based on zaslavskii was

used in [11]. But in [14], PSO was hybridized with chaotic local search procedure based on logistic map. The Logistic and Tent chaotic maps were, respectively, used as inertial weight by [9] in binary PSO to handle feature selection problem. In [12], twelve different chaos maps were implemented to tune the attraction parameter of accelerated PSO algorithm.

The aim of this paper is to further investigate the performances of two PSO variants, LDIW-PSO and RIW-PSO algorithms, with various chaotic maps incorporated into their IWSs. For this purpose, 9 additional chaotic maps along with logistic map are introduced in this paper and used with the two variants at different times. Some well-studied benchmark mathematical problems in the literature were used to test the algorithms using these maps. The outcome of the experiments should help ascertain the chaotic maps that contribute better to the performances of the algorithms in comparison to logistic chaotic map in order to provide some useful information regarding the usage of these maps in the IWSs of the PSO variants.

In the sections that follow, the inertia weight PSO and its variants considered in this paper are introduced in Section 2, chaotic maps used in the experiments are described in Section 3, setting of the experiments is given in Section 4, and experimental results and discussions are presented in Section 5, while Section 6 concludes the paper.

2. Inertia Weight PSO

PSO algorithm is a population-based evolutionary stochastic technique made up of a swarm of particles which coexist and evolve simultaneously based on knowledge shared with neighbouring particles. The PSO process is initialized with a swarm of random particles in the search space and the algorithm is allowed to execute a number of times in order to carry out a search for optimal solutions in the search space. In inertia weight PSO, each particle is assumed to have position and velocity in a physical n-dimensional search space; the position and velocity of a particle i in each iteration t is represented as the vectors $\vec{X}_i = (x_{i1}, \ldots, x_{\text{in}})$ and $\vec{V}_i = (v_{i1}, \ldots, v_{\text{in}})$, respectively. When the particles move in the search space searching for the optimum solution for a particular optimization problem, other particles follow the current optimum particle by adjusting their velocities and positions using (1). The positions and velocities of the particles are confined within $[X_{\min}, X_{\max}]^n$ and $[V_{\min}, V_{\max}]^n$, respectively, as follows:

$$\vec{V}_i^{t+1} = \omega \vec{V}_i^t + c_1 \vec{r}_1 \left(\overrightarrow{p\text{best}}_i^t - \vec{X}_i^t \right) + c_2 \vec{r}_2 \left(\overrightarrow{g\text{best}}^t - \vec{X}_i^t \right), \qquad \vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1}. \tag{1}$$

A particle's position is taken as possible solution for the problem being optimized, while the fitness of this possible solution is determined by evaluating the problem's objective function. The best position searched by the particle itself so far ($\overrightarrow{p\text{best}}_i^t$) and the optimization position searched by the whole particle swarms so far ($\overrightarrow{g\text{best}}^t$) are n-dimensional vectors representing personal best position of particle i at iteration t and global best positions selected from the personal best positions of all the particles in the swarm at iteration t. whereas $\vec{r}_1$ and $\vec{r}_2$ are two n-dimensional vectors of random numbers between 0 and 1, which introduces randomness to the searching strategy, and the two positive constants c_1 and c_2 are cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of $\overrightarrow{p\text{best}}_i^t$ and $\overrightarrow{g\text{best}}^t$.

The inertia weight (ω) strikes a balance between exploration and exploitation characteristics of PSO and it determines the level of contribution of previous particle velocity to the present velocity.

2.1. Linear Decreasing Inertia Weight PSO (LDIW-PSO). This variant implements the linear decreasing IWS which has greatly improved the algorithm [15, 16]. In this variant, the inertia weight starts with a large initial value and then linearly decreases to a smaller final value with the belief that a large inertia weight facilitates a global search, while a small inertia weight facilitates a local search. The commonly used initial and final values are 0.9 and 0.4 [10, 17, 18]; other values have also been used [19–21]. The inertia weight creates a means of flexibility for the movements of particles in the search space. According to [22], relatively high inertia weight value (e.g., 0.9) creates a medium of low viscosity for the particles to facilitate extensive exploration while gradually reducing it to a much lower value (e.g., 0.4) creates a high viscosity medium to facilitate exploitation. The experimental results in [15] showed that using the linearly decreasing inertia weight can make PSO suffer from premature convergence due to lack of search ability towards the end of run to jump out of the local minimum in some cases. Because of this challenge, employing adapting strategy for adjusting the inertia weight was suggested to improve the performance PSO near the optima [15]. Many researchers have made efforts to achieve this through various improvements on LDIW-PSO by introducing new parameters into its IWS or proposing new IWS to generally improve the performance of PSO technique [5, 10, 17, 19, 23, 24]. Equation (2) represents the commonly used LDIW strategy:

$$\omega_t = (\omega_{\text{start}} - \omega_{\text{end}}) \left(\frac{T_{\max} - t}{T_{\max}} \right) + \omega_{\text{end}}, \tag{2}$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, t is the current iteration number, $T_{\max}$ is the maximum iteration number, and $\omega_t \in [0, 1]$ is the inertia weight value in the tth iteration.

2.2. Random Inertia Weight PSO (RIW-PSO). There are different IWSs with random features [5–8], but the one in [5] is adopted in this paper for the experiments. With the aim of using particle swarms to track and optimize dynamic systems, a new way of calculating the inertia weight value was proposed in [5] as shown in (3). The formula makes the inertia weight change randomly and produces a number

randomly varying between 0.5 and 1.0, with a mean value of 0.75. Consider

$$\omega_t = 0.5 + \frac{\text{rand}(\cdot)}{2}. \quad (3)$$

In (3), rand(·) is a uniformly distributed random number within the range $[0, 1]$. As a result of the difficulty in predicting whether exploration (a larger inertia weight value) or exploitation (a smaller inertia weight) will be better at any given time in tracking a nonlinear dynamic system, the strategy in (3) was introduced by [5] to address the inefficiency of linearly decreasing inertia weight, which decreases from 0.9 to 0.4 during a run, in handling such a problem.

2.3. Chaotic Inertia Weight PSO. The chaotic inertia weights PSO that were proposed in [10] are Chaotic Decreasing Inertia Weight PSO (CDIW-PSO) and Chaotic Random Inertia Weight PSO (CRIW-PSO) shown in (5) and (6), respectively. The aim was to improve LDIW-PSO and RIW-PSO using logistic map in order to avoid getting into local optimum in searching process by utilizing the merits of chaotic optimization. Logistic map, represented in (4), is one of the simplest maps that appear in nonlinear dynamics of biological population evidencing chaotic behavior. Consider

$$z_{k+1} = \mu z_k (1 - z_k), \quad (4)$$

where k is the iteration number, x_k is the kth chaotic number, and $\mu = 4.0$. This map generates values between 0 and 1, provided that the initial value $z_0 \in (0, 1)$ and that $z_0 \notin (0.0, 0.25, 0.5, 0.75, \text{ and } 1.0)$.

$$\omega_t = (\omega_{\text{start}} - \omega_{\text{end}}) \left(\frac{(T_{\max} - t)}{T_{\max}} \right) + \omega_{\text{end}} \times z_{k+1}, \quad (5)$$

$$\omega_t = 0.5 \times \text{rand}(\cdot) + 0.5 \times z_{k+1}, \quad (6)$$

where ω_{start} and ω_{end} are the initial and final values of inertia weight, rand(·) is a uniform random number in $[0, 1]$, t is the current iteration, and $T_{\max}$ is the maximum iteration. The results in [10] show that the PSO had preferable convergence precision, quick convergence velocity, and better global search ability. This is because, due to nonrepetition of chaos, the algorithm could carry out overall searches at higher speed, diversify the particles, and improve the algorithm's performance in preventing premature convergence too quickly to local minima compared with RIW-PSO and LDIW-PSO which have no chaos characteristics. No other chaotic maps were implemented with the linear decreasing and random IWSs to see if they could make the algorithms perform better in comparison to using logistic map. Besides, the results in [10] leave much room for further improvement on the performance of RIW-PSO and LDIW-PSO.

3. Chaotic Maps

Chaos is a deterministic dynamic system that is very sensitive to its initial conditions and parameters. The application of chaotic maps instead of random sequence in PSO is a powerful strategy to diversify the swarm and improve its performance. There are various chaotic maps that exist in the literature which could also be used with RIW-PSO and LDIW-PSO apart from logistic map used in [10]. Introduced below are some of these maps, adopted from [2, 12], and they are described as used in the experiments conducted in this paper.

3.1. Circle. This map has two parameters a (which can be interpreted as the strength of nonlinearity) and b (which can be interpreted as externally applied frequency). It is a one-dimensional map which maps a Circle into itself and it is represented by (7):

$$x_{k+1} = \left(x_k + b - \frac{a}{2\pi} \sin(2\pi x_k) \right) \bmod (1). \quad (7)$$

In this paper, $a = 0.5$ and $b = 0.2$.

3.2. Cubic. This map is somehow similar to Sine map, but it generates values in the interval $[-1.5, 1.5]$. It is defined as

$$x_{k+1} = 3x_k \left(1 - x_k^2\right). \quad (8)$$

In this paper, the values generated were normalized between 0 and 1.

3.3. Gaussian. This map is also known as Gauss or mouse map. It is defined as

$$x_{k+1} = \begin{cases} 0, & x_k = 0 \\ \dfrac{1}{x_k} \bmod (1), & x_k \in (0, 1), \end{cases} \quad (9)$$

where $(1/x_k) \bmod (1) = (1/x_k) - \lfloor 1/x_k \rfloor$ and $\lfloor z \rfloor$ denotes the largest integer less than z which acts as a shift on the continued fraction representation of numbers.

3.4. Intermittency. This map is the extension of the Bernoulli Shift in which one of the piecewise linear segments is replaced by a nonlinear segment as shown below

$$x_{k+1} = \begin{cases} \varepsilon + x_k + c x_k^m, & x_k \in (0, d] \\ \dfrac{(x_k - d)}{(1 - d)}, & x_k \in (d, 1), \end{cases} \quad (10)$$

where $\varepsilon = 10^{-3}$, $m = 2$, $d = 0.7$, and $c = (1 - \varepsilon - d)/d^m$.

3.5. Iterative Chaotic Map with Infinite Collapses (ICMIC). This map is represented by

$$x_{k+1} = \sin\left(\frac{a\pi}{x_k} \right), \quad (11)$$

where $a(0, 1)$, $a = 0.85$ and the results were normalized between 0 and 1.

3.6. Piecewise. This map consists of line segments and is defined as

$$x_{k+1} = \begin{cases} \dfrac{x_k}{p}, & x_k \in [0, p) \\ \dfrac{(x_k - p)}{(0.5 - p)}, & x_k \in [p, 0.5) \\ \dfrac{(1 - p - x_k)}{(0.5 - p)}, & x_k \in [0.5, 1 - p) \\ \dfrac{(1 - x_k)}{p}, & x_k \in [1 - p, 1), \end{cases} \tag{12}$$

where p is the control parameter between 0 and 0.5.

3.7. Sinusoidal. This map is represented by the following

$$x_{k+1} = \beta \sin(\pi x_k). \tag{13}$$

This map is similar to logistic map in shape with $\beta = 1$ and generates values in $(0, 1)$. The values were normalized between 0 and 1 for $\beta > 1$.

3.8. Skew Tent. This map is a Tent map skewed to either left or right controlled by the parameter p. The map is defined by

$$x_{k+1} = \begin{cases} \dfrac{x_k}{p}, & x_k \in [0, p) \\ \dfrac{(1 - x_k)}{(1 - p)}, & x_k \in [p, 1], \end{cases} \tag{14}$$

where $p = 0.3$.

3.9. Tent. The tent map is like the logistic map but has a "∧" shape unlike logistic map which has a dome-like shape. It is defined by

$$x_{k+1} = \begin{cases} 2px_k, & x_k \in [0, 0.5] \\ 2p(1 - x_k), & x_k \in (0.5, 1], \end{cases} \tag{15}$$

where k is the iteration number, x_k is the kth chaotic number and $p = 0.99$. This map also generates values between 0 and 1, provided that the initial value $x_0 \in [0, 1]$.

4. Experimental Setup

To investigate the performance of RIW-PSO and LDIW-PSO with the chaotic maps incorporated into them at different times, five (5) well-studied benchmark problems described below were used to validate them. The swarm size was set to 20 particles, while the number of variables (dimensions) for all problems was set to 30 and 50 with respective maximum number of iterations of 2000 and 3000. The algorithm was allowed to run the maximum number of iterations and number of successful runs recorded. A run is successful if the mean fitness value obtained by an algorithm is less than the success criterion after the maximum iteration. Values for ω_{start} and ω_{end} were set to 0.9 and 0.4 and c_1 and c_2 were set to 2.0 as used in [10] and $V_{\max}$ was clamped to be 15% of the search space [25]. For fairness, the same random seeds were used in all the experiments with 50 independent runs for the test problems. The performance criteria were the mean best solution, standard deviation, success rate, and number of function evaluations of the algorithms. The simulation program was developed in Microsoft Visual C# programming language.

The success rate (SR) was computed according to (16) and expected number of function evaluations (NFE) according to (17):

$$\text{SR} = \frac{G_{\text{times}}}{T_{\text{runs}}}, \tag{16}$$

where G_{times} is the total number of times the set goal was reached over 50 independent runs and T_{runs} is the total number of independent runs (i.e., 50, in our experiment):

$$\text{NFE} = S_{\text{size}} \times \frac{t_{\text{avg}}}{\text{SR}}, \tag{17}$$

where S_{size} is the swarm size and t_{avg} is the average number of iterations the set goal was reached over 50 independent runs.

4.1. Test Problems. The test problems used in the experiments are well studied in the literature [26–29]. They are Ackley, Griewank, Rastrigin, Rosenbrock, and Sphere. All the problems are continuous, scalable, and multimodal except the last two problems, which are unimodal. Each of these problems is described below.

The *Ackley* problem is nonseparable. It is a widely used test problem and it is defined in (18). The global minimum $f_1(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$ and the number of local minima is not known. Consider

$$f_1(\vec{x}) = -20 \exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d} \cos(2\pi x_i)\right) + 20 + e. \tag{18}$$

The *Griewank* problem is non-separable with many widespread local minima regularly distributed. The complexity of the problem decreases as the dimensionality increases. Its global minimum $f_2(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$ and the number of local minima in a two-dimensional case is about 500. This problem is represented by

$$f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1. \tag{19}$$

The *Rastrigin* problem represented in (20) is separable and has many local minima arranged in a lattice-like configuration. It is based on the Sphere problem with the addition of cosine modulation so as to produce frequent local minima. There are about 50 local minima for two-dimensional case

Table 1: Search ranges, optimal values, and success criteria for the test problems.

	f_1	f_2	f_3	f_4	f_5
Search range	$[-30, 30]$	$[-600, 600]$	$[-5.12, 5.12]$	$[-30, 30]$	$[-100, 100]$
Optimal value	0	0	0	0	0
Success criteria	0.01	0.05	50.0	100.0	0.01

and its global minimum $f_3(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$. Consider

$$f_3(\vec{x}) = \sum_{i=1}^{d} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right). \tag{20}$$

Shown in (21) is the *Rosenbrock* problem; it is non-separable. It is a classic optimization problem also known as banana function, the second function of De Jong, or extended Rosenbrock function. Its global minimum $f_4(\vec{x}) = 0$ obtainable at $\vec{x} = 1$, lies inside a long narrow, parabolic's shaped valley. Though it look-simple to solve, yet due to a saddle points it is very difficult to converge to the global optimum. Consider

$$f_4(\vec{x}) = \sum_{i=1}^{d-1} \left(100\left(x_{i+1} - x_i^2\right)^2\right) + \left(x_i - 1\right)^2. \tag{21}$$

The *Sphere* problem also known as the first De Jong function is separable. It is one of the simplest test benchmark problems. Its global minimum $f_6(\vec{x}) = 0$ is obtainable at $\vec{x} = 0$ and the problem is represented by

$$f_6(\vec{x}) = \sum_{i=1}^{d} x_i^2. \tag{22}$$

4.2. Properties of the Test Problems. Shown in Table 1 are the properties of the test problems. The success criteria are stated here as used in [10].

5. Results and Discussions

Many researchers have measured the performance of PSO algorithms using mean (average) fitness value and standard deviation [17, 19, 30–35]. The number of iterations or evaluations of the objective function that takes the algorithm to find optimum solution with specified accuracy to an optimization problem has also been used. However, the number of function evaluations appears to be more informative and popularly used to measure the performance of optimization algorithms [3, 25]. This is because it reflects the time or computational complexity of optimization algorithms and takes into account the swarm size, average number of algorithm iterations to reach the set goal (e.g., success threshold), and the success rate of the algorithm. In this paper, average fitness value, standard deviation of fitness value, success rate, and number of function evaluations were used for the performance measurement of the two PSO variants.

The mean best fitness (mean) is a measure of the precision that an algorithm can get within a given number of iterations; standard deviation (SD) is a measure of the algorithm's stability and robustness, while success rate (SR) is the number of times an algorithm is able to meet success criterion out of a specified number of independent runs, which is a reflection of global search ability of the algorithm.

Tables 2–11 show the numerical results obtained in the experiments when the various chaotic maps were incorporated into RIW-PSO and LDIW-PSO algorithms. In the tables, "None" means that the algorithms were implemented without any chaotic map. In Tables 5 and 10, "—" indicates that no trial run satisfied the success criterion. The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others. The values with asterik $(*)$ are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

5.1. Results for LDIW-PSO Using the Various Chaotic Maps. Presented in Tables 2–5 are the results with respect to mean fitness value, standard deviation, success rate, and number of function evaluations as obtained in the experiments.

Table 2 shows the mean fitness values obtained by LDIW-PSO using the various chaotic maps. None of the chaotic maps used with the variants could enable it to perform best in all test problems compared with other maps. Apart from Sine, Intermittency and Cubic maps, the algorithm was able to obtain best optimal fitness for different test problems using other maps. Circle and Gaussian maps look more robust than others, because with them the algorithm was able to obtain the best optimal fitness for two test problems in the two problem dimensions which shows better characteristics for convergence precision. Besides f_3, logistic map was outperformed by other maps as shown by the shaded portions in obtaining better optimal fitness.

Table 3 shows the standard deviation (stability measure) of LDIW-PSO using the various chaotic maps. As shown by the results, none of the chaotic maps when used with the variant could enable the algorithm to have the best stability across the test problems. However, the algorithm achieved the best stabilities with the various maps for different test problems, except with Sine, Intermittency, and Cubic maps. The algorithm looks more stable using Piecewise and Gaussian maps than others, because with them it was able to obtain the best stability in two test problems with the two problem dimensions. Other maps were able to facilitate better stability than logistic map as shown by the shaded portions in the table.

Table 4 shows the global search ability of LDIW-PSO using the various chaotic maps. The algorithm had the same search ability using all the maps for f_5 in both dimensions

Table 2: Mean fitness obtained by LDIW-PSO for test problems using the various chaotic maps.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	**1.715e − 05**	1.896e − 02	4.563e + 01	1.922e + 03	4.797e − 09	2.060e − 01	8.604e − 03	9.091e + 01	5.542e + 03	2.753e − 05
Logistic	3.798e − 02	1.549e − 02	**4.064e + 01**	4.698e + 01	4.109e − 20	1.701e − 01	1.620e − 02	8.875e + 01	9.197e + 01	6.010e − 13
Tent	2.680e − 02*	1.403e − 02*	4.579e + 01	3.764e + 01*	4.505e − 21*	2.608e − 01	1.279e − 02*	8.716e + 01*	**8.592e + 01***	4.108e − 13*
Skew Tent	3.724e − 02*	2.305e − 02	4.172e + 01	1.839e + 03	1.270e − 22*	1.299e − 01*	1.785e − 02	**8.184e + 01***	1.578e + 02	1.028e − 13*
Sine	3.002e − 02*	1.602e − 02	4.080e + 01	4.219e + 01*	1.792e − 21*	2.011e − 01	1.396e − 02*	8.971e + 01	1.915e + 03	1.020e − 10
ICMIC	7.621e − 02	**1.151e − 02***	4.239e + 01	3.709e + 01*	2.422e − 19	2.365e − 01	1.668e − 02	8.646e + 01*	1.955e + 03	2.478e − 11
Circle	4.171e − 02	1.753e − 02	4.396e + 01	**3.524e + 01**	8.206e − 23*	2.970e − 01	**7.969e − 03***	8.562e + 01*	9.215e + 01	2.346e − 13*
Piecewise	5.601e − 02	1.412e − 02*	4.116e + 01	5.026e + 01	1.069e − 21*	**8.191e − 02***	1.256e − 02*	8.799e + 01*	2.728e + 02	9.247e − 14*
Gaussian	6.033e − 02	1.597e − 02	4.367e + 01	4.487e + 01*	**1.553e − 23***	1.550e − 01*	1.404e − 02*	8.234e + 01*	1.005e + 02	**8.369e − 15***
Intermittency	1.128e − 01	1.421e − 02*	4.311e + 01	3.717e + 01*	1.801e − 23*	4.249e − 01	1.344e − 02*	8.454e + 01*	1.503e + 02	3.636e − 14*
Cubic	2.309e − 02*	1.636e − 02	4.230e + 01	6.200e + 01	9.412e − 21*	1.707e − 01	1.449e − 02*	8.585e + 01*	8.703e + 01*	9.514e − 13

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
* are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 3: Standard deviation for test function using LDIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	**2.312e − 05**	2.095e − 02	1.226e + 01	1.259e + 04	8.589e − 09	5.041e − 01	1.292e − 02	1.737e + 01	2.136e + 04	9.913e − 05
Logistic	2.659e − 01	2.020e − 02	9.595e + 00	3.596e + 01	1.592e − 19	4.681e − 01	2.309e − 02	1.879e + 01	4.125e + 01	2.656e − 12
Tent	1.876e − 01*	1.539e − 02*	1.322e + 01	3.158e + 01*	1.717e − 20*	5.704e − 01	1.960e − 02*	2.063e + 01	**3.599e + 01***	1.614e − 12*
Skew Tent	1.824e − 01*	2.854e − 02	**7.706e + 00***	1.260e + 04	2.713e − 22*	3.938e − 01*	2.635e − 02	**1.492e + 01***	4.246e + 02	3.658e − 13*
Sine	2.102e − 01*	2.096e − 02	1.289e + 01	4.773e + 01	5.897e − 21*	5.112e − 01	2.783e − 02	2.032e + 01	1.260e + 04	7.103e − 10
ICMIC	3.043e − 01	1.359e − 02*	1.169e + 01	**2.997e + 01***	1.680e − 18	5.112e − 01	2.483e − 02	2.198e + 01	1.260e + 04	1.722e − 10
Circle	2.056e − 01*	1.997e − 02*	1.328e + 01	3.278e + 01*	2.914e − 22*	5.665e − 01	**1.257e − 02***	1.732e + 01*	5.013e + 01	1.572e − 12*
Piecewise	2.787e − 01	**1.271e − 02***	1.294e + 01	3.422e + 01*	3.728e − 21*	**3.263e − 01***	1.767e − 02*	2.446e + 01	7.173e + 02	2.610e − 13*
Gaussian	2.402e − 01*	1.916e − 02*	1.296e + 01	3.288e + 01*	**3.986e − 23***	4.304e − 01*	1.903e − 02*	1.976e + 01	8.620e + 01	**2.227e − 14***
Intermittency	3.861e − 01	1.817e − 02*	1.126e + 01	3.089e + 01*	7.894e − 23*	6.989e − 01	2.052e − 02*	1.633e + 01*	4.260e + 02	2.427e − 13*
Cubic	1.617e − 01*	1.760e − 02*	1.290e + 01	8.995e + 01	4.637e − 20*	4.679e − 01*	1.791e − 02*	1.910e + 01	4.661e + 01	5.196e − 12

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
* are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 4: Success rate for test problems using LDIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	**50/50**	25/50	32/50	38/50	50/50	37/50	34/50	1/50	17/50	50/50
Logistic	49/50	**29/50**	42/50	45/50	50/50	44/50	27/50	0/50	36/50	50/50
Tent	49/50	25/50	33/50	**49/50***	50/50	41/50	30/50*	1/50*	**41/50***	50/50
Skew Tent	48/50	23/50	**43/50***	48/50*	50/50	45/50*	27/50	1/50*	30/50	50/50
Sine	49/50	25/50	38/50	48/50*	50/50	43/50	35/50*	1/50*	31/50	50/50
ICMIC	47/50	**29/50**	39/50	47/50*	50/50	41/50	28/50*	1/50*	34/50	50/50
Circle	48/50	24/50	36/50	48/50*	50/50	39/50	**36/50***	0/50	37/50*	50/50
Piecewise	48/50	25/50	37/50	46/50*	50/50	**47/50***	32/50*	1/50*	34/50	50/50
Gaussian	47/50	**29/50**	36/50	48/50*	50/50	44/50	31/50*	1/50*	37/50*	50/50
Intermittency	46/50	25/50	38/50	**49/50***	50/50	36/50	33/50*	0/50	36/50	50/50
Cubic	49/50	25/50	42/50	46/50*	50/50	44/50	29/50*	0/50	36/50	50/50

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
*are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

and performed poorly in f_3 under dimension 50. From the results, the algorithm seems to have better global search ability using Tent map than other maps, because with the map it had the best search ability in both problem dimensions for f_4. However, the algorithm could not demonstrate the best global search ability across the test problems and dimensions using any of the chaotic maps. Sine and Cubic maps had least positive influence on the algorithm in terms of search ability. Besides f_2, other maps had better influence on the algorithm in achieving better global search ability than logistic map as shown by the shaded portions in the table.

Table 5 shows the number of function evaluations by the algorithm, using the various chaos maps. When Intermittency map was used and the problem dimension was set to 30, the algorithm had the lowest number of function evaluations in all the test problems except in two of the test problems under dimension 50. As indicated by the shaded portions, the algorithm executed lesser number of function evaluations using other chaotic maps than logistic map.

Table 6 shows the average ranking of the performance of LDIW-PSO when each of the chaotic maps was incorporated into it to solve all the test problems. In other words, each value in the table represents the average rank of the corresponding map in comparison to others across the test problems for each problem dimension. The least value indicates that the associated map performed best, while the largest value indicates that the associated map performed worst. Generally, when the problem dimension was set to 30, the algorithm obtained the best convergence precision using Sine map and was more stable using Intermittency map. But it demonstrated the best global search ability with Logistic, Skew Tent, Sine, and cubic maps. Less computational effort was needed using Intermittency map compared with others. When the problem dimension was set to 50, the algorithm obtained the best convergence precision and was more stable when Gaussian map was used. But it demonstrated the best global search ability using Circle and Piecewise maps. Less computational effort was needed using Gaussian and Intermittency maps in comparison with others. On the average, Intermittency map performed best when the problem dimension was 30, while Gaussian map performed best when the problem dimension was 50.

5.2. Results for RIW-PSO Using the Various Chaotic Maps. Presented in Tables 7–10 are the results with respect to mean fitness value, standard deviation, success rate, and number of function evaluations as obtained in the experiments by RIW-PSO.

Table 7 shows the mean fitness values obtained by RIW-PSO using the various chaotic maps. None of the chaotic maps could make the algorithm perform the best across the test problems in both dimensions. Apart from Skew Tent, Sine, and Gaussian maps, the algorithm was able to obtain best optimal fitness for different test problems using other maps. Intermittency map looks more effective than others, because with it the algorithm was able to obtain best optimal fitness for three test problems in the two problem dimensions showing better characteristics for convergence precision. Apart from f_2, logistic map was less effective compared with other maps as shown by the shaded portions.

Presented in Table 8 is the standard deviation (stability measure) obtained by RIW-PSO using the various chaotic maps. From the results, none of the chaotic maps could make the algorithm have the best stability across the test problems. However, the algorithm achieved the best stabilities with the various maps for different test problems across the problem dimensions, except with ICMIC, Gaussian, and Cubic maps. The algorithm looks more stable using Logistic, Tent and intermittency maps than others, because with them it was able to obtain the best stability in two test problems. Besides f_2 and f_3 with dimension 30, other maps were able to facilitate better stability than logistic map as shown by the shaded portions in the table.

Table 9 shows the global search ability of RIW-PSO using the various chaotic maps. The algorithm had the same search ability using all the maps for f_5 in both dimensions. As shown in the results, the algorithm seems to have better global search ability using Logistic, Tent, Sine and Cubic maps than other

Table 5: Number of function evaluations for test problems using LDIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	$3.203e+04$	$6.317e+04$	$3.803e+04$	$4.125e+04$	$3.016e+04$	$7.468e+04$	$7.705e+04$	$2.315e+06$	$1.598e+05$	$5.144e+04$
Logistic	$1.867e+04$	$3.195e+04$	$1.096e+04$	$1.868e+04$	$1.627e+04$	$3.987e+04$	$5.786e+04$	—	$5.631e+04$	$3.032e+04$
Tent	$1.846e+04^*$	$3.519e+04$	$1.484e+04$	$1.770e+04^*$	$1.643e+04$	$4.464e+04$	$5.282e+04^*$	$\mathbf{9.860e+05}^*$	$5.003e+04^*$	$3.068e+04$
Skew Tent	$1.828e+04^*$	$3.649e+04$	$1.192e+04$	$1.541e+04^*$	$1.429e+04^*$	$3.835e+04^*$	$5.522e+04^*$	$1.207e+06^*$	$6.886e+04$	$2.953e+04^*$
Sine	$1.904e+04$	$3.486e+04$	$1.171e+04$	$1.732e+04^*$	$1.622e+04^*$	$4.173e+04$	$4.582e+04^*$	$1.390e+06^*$	$6.809e+04$	$3.115e+04$
ICMIC	$1.898e+04$	$3.028e+04^*$	$1.219e+04$	$1.664e+04^*$	$1.603e+04^*$	$4.265e+04$	$5.522e+04^*$	$1.261e+06^*$	$5.808e+04$	$3.028e+04^*$
Circle	$1.624e+04^*$	$3.065e+04^*$	$9.328e+03^*$	$1.425e+04^*$	$1.346e+04^*$	$3.987e+04$	$\mathbf{3.807e+04}^*$	—	$4.920e+04^*$	$2.627e+04^*$
Piecewise	$1.837e+04^*$	$3.501e+04$	$1.245e+04$	$1.885e+04$	$1.581e+04^*$	$\mathbf{3.629e+04}^*$	$4.857e+04^*$	$1.095e+06^*$	$6.127e+04$	$3.010e+04^*$
Gaussian	$1.765e+04^*$	$2.787e+04^*$	$1.155e+04$	$1.534e+04^*$	$1.459e+04^*$	$3.700e+04^*$	$4.624e+04^*$	$1.142e+06^*$	$5.067e+04^*$	$2.780e+04^*$
Intermittency	$\mathbf{1.537e+04}^*$	$\mathbf{2.673e+04}^*$	$\mathbf{8.369e+03}^*$	$\mathbf{1.277e+04}^*$	$\mathbf{1.221e+04}^*$	$3.949e+04^*$	$3.813e+04^*$	—	$\mathbf{4.758e+04}^*$	$\mathbf{2.415e+04}^*$
Cubic	$1.852e+04^*$	$3.520e+04$	$1.199e+04$	$1.850e+04^*$	$1.620e+04^*$	$4.188e+04$	$5.488e+04^*$	—	$5.633e+04$	$3.086e+04$

—Indicates that no trial run satisfied the success criterion.
The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
*are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 6: Average ranking of the performance of LDIW-PSO relative to the various chaotic maps.

Chaos maps	Problem dimension = 30					Problem dimension = 50				
	Mean fitness	Standard deviation	Success rate	Function evaluation	Average performance	Mean fitness	Standard deviation	Success rate	Function evaluation	Average performance
None	8.6	7.2	3.6	11	7.60	8.4	6.8	3.8	10.2	7.30
Logistic	5.6	6.8	**2.2**	6.8	5.35	6.4	5.4	3.8	7	5.65
Tent	5.4	5.4	2.6	8.2	5.40	5.4	6.4	3	5.6	5.10
Skew Tent	6.8	5.8	**2.2**	5.4	5.05	5	4.6	4	5.8	4.85
Sine	**4.8**	7.2	**2.2**	7	5.30	8.2	9	2.6	7	6.70
ICMIC	5.8	5.4	2.4	6.2	4.95	8.4	9.2	4	7.2	7.20
Circle	5.8	6.2	3	2.4	4.35	4.8	4.4	**2.4**	3.8	3.85
Piecewise	5.4	5.8	3	7.2	5.35	4.6	5.2	**2.4**	4.2	4.10
Gaussian	6	5.6	2.8	3.2	4.40	**3.6**	**4.2**	2.6	**3.2**	**3.40**
Intermittency	5.4	**4.6**	2.6	**1**	**3.40**	5.4	5.8	3.6	**3.2**	4.50
Cubic	6.4	6	**2.2**	7.6	5.55	5.6	5	3.4	7.6	5.40

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.

Table 7: Mean fitness obtained by RIW-PSO for test problems using the various chaotic maps.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	$3.358e+00$	$1.496e+00$	$6.988e+01$	$4.173e+03$	$5.512e+01$	$5.022e+00$	$4.905e+00$	$1.884e+02$	$3.062e+04$	$4.352e+02$
Logistic	$8.915e-07$	$\mathbf{1.411e-02}$	$3.933e+01$	$5.085e+01$	$5.404e-13$	$9.938e-04$	$1.303e-02$	$7.041e+01$	$1.218e+02$	$2.798e-07$
Tent	$3.712e-07^*$	$1.610e-02$	$4.555e+01$	$5.092e+01$	$2.099e-13^*$	$\mathbf{1.701e-04}^*$	$1.007e-02^*$	$7.237e+01$	$1.344e+02$	$1.574e-08^*$
Skew Tent	$2.309e-02$	$2.055e-02$	$4.267e+01$	$4.710e+01^*$	$3.280e-15^*$	$3.468e-02$	$2.172e-02$	$7.044e+01$	$1.161e+02^*$	$1.443e-09^*$
Sine	$5.360e-02$	$1.496e-02$	$3.877e+01^*$	$5.341e+01$	$3.498e-13^*$	$4.281e-04^*$	$1.367e-02$	$6.922e+01^*$	$1.171e+02^*$	$7.386e-08^*$
ICMIC	$3.798e-02$	$1.450e-02$	$4.252e+01$	$6.736e+01$	$9.963e-15^*$	$2.623e-04^*$	$1.885e-02$	$\mathbf{6.894e+01}^*$	$1.138e+02^*$	$8.323e-09^*$
Circle	$5.360e-02$	$1.577e-02$	$4.128e+01$	$1.210e+02$	$\mathbf{1.167e-17}^*$	$2.772e-02$	$1.985e-02$	$7.641e+01$	$9.886e+01^*$	$2.253e-10^*$
Piecewise	$\mathbf{1.913e-07}^*$	$1.686e-02$	$4.096e+01$	$5.357e+01$	$2.019e-14^*$	$7.353e-04^*$	$\mathbf{9.627e-03}^*$	$7.256e+01$	$1.125e+02^*$	$5.960e-09^*$
Gaussian	$3.002e-02$	$1.825e-02$	$4.248e+01$	$5.155e+01$	$3.693e-17^*$	$2.950e-02$	$2.072e-02$	$7.188e+01$	$1.047e+02^*$	$7.502e-11^*$
Intermittency	$5.674e-01$	$1.541e-02$	$4.239e+01$	$\mathbf{4.453e+01}^*$	$1.357e-16^*$	$3.831e-01$	$2.915e-02$	$7.362e+01$	$\mathbf{9.456e+01}^*$	$\mathbf{3.699e-13}^*$
Cubic	$4.629e-07^*$	$1.905e-02$	$\mathbf{3.824e+01}^*$	$1.095e+02$	$2.368e-13^*$	$4.757e-04^*$	$1.457e-02$	$6.957e+01^*$	$1.785e+02$	$7.570e-08^*$

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
*are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 8: Standard deviation for test problems using RIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	$4.495e-01$	$2.886e-01$	$1.917e+01$	$1.248e+04$	$3.207e+01$	$4.505e-01$	$1.330e+00$	$3.031e+01$	$1.589e+04$	$1.488e+02$
Logistic	$1.755e-06$	$\mathbf{1.552e-02}$	$\mathbf{1.045e+01}$	$4.145e+01$	$1.308e-12$	$1.398e-03$	$1.946e-02$	$1.724e+01$	$9.216e+01$	$6.141e-07$
Tent	$1.476e-06^*$	$2.103e-02$	$1.265e+01$	$3.762e+01^*$	$1.205e-12^*$	$\mathbf{3.402e-04}^*$	$\mathbf{1.536e-02}^*$	$1.547e+01^*$	$8.253e+01^*$	$2.449e-08^*$
Skew Tent	$1.617e-01$	$2.361e-02$	$1.172e+01$	$4.014e+01^*$	$1.027e-14^*$	$2.416e-01$	$3.313e-02$	$\mathbf{1.435e+01}^*$	$4.498e+01^*$	$3.137e-09^*$
Sine	$2.626e-01$	$2.184e-02$	$1.091e+01$	$\mathbf{3.276e+01}^*$	$1.089e-12^*$	$5.928e-04^*$	$2.240e-02$	$1.648e+01^*$	$9.425e+01$	$1.251e-07^*$
ICMIC	$2.659e-01$	$1.924e-02$	$1.090e+01$	$8.112e+01$	$3.054e-14^*$	$7.182e-04^*$	$3.112e-02$	$2.289e+01$	$6.810e+01^*$	$3.330e-08^*$
Circle	$2.626e-01$	$1.598e-02$	$1.190e+01$	$4.240e+02$	$\mathbf{2.197e-17}^*$	$1.923e-01$	$3.271e-02$	$2.371e+01$	$5.792e+01^*$	$7.925e-10^*$
Piecewise	$\mathbf{7.527e-07}^*$	$2.486e-02$	$1.159e+01$	$4.126e+01^*$	$8.114e-14^*$	$3.018e-03$	$1.540e-02^*$	$1.737e+01$	$4.794e+01^*$	$1.365e-08^*$
Gaussian	$2.102e-01$	$2.749e-02$	$1.342e+01$	$3.686e+01^*$	$6.254e-17^*$	$2.059e-01$	$3.026e-02$	$1.650e+01^*$	$7.298e+01^*$	$1.098e-10^*$
Intermittency	$8.331e-01$	$1.872e-02$	$1.420e+01$	$3.717e+01^*$	$9.441e-16^*$	$6.750e-01$	$4.043e-02$	$1.999e+01$	$\mathbf{4.109e+01}^*$	$\mathbf{1.555e-12}^*$
Cubic	$1.111e-06^*$	$2.601e-02$	$1.190e+01$	$4.211e+02$	$8.107e-13^*$	$8.193e-04^*$	$2.117e-02$	$1.807e+01$	$4.215e+02$	$1.507e-07^*$

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
* are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 9: Success rate for test function using RIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	0/50	0/50	7/50	0/50	0/50	0/50	0/50	0/50	0/50	0/50
Logistic	**50/50**	26/50	**43/50**	46/50	50/50	**50/50**	29/50	4/50	27/50	50/50
Tent	**50/50**	29/50*	36/50	47/50*	50/50	**50/50**	34/50*	2/50	21/50	50/50
Skew Tent	49/50	21/50	41/50	48/50*	50/50	49/50	29/50	2/50	23/50	50/50
Sine	48/50	26/50	40/50	**49/50***	50/50	**50/50**	33/50*	6/50*	29/50*	50/50
ICMIC	49/50	**30/50***	39/50	47/50*	50/50	**50/50**	26/50	8/50*	30/50*	50/50
Circle	48/50	26/50	37/50	45/50	50/50	49/50	29/50	**10/50***	31/50*	50/50
Piecewise	**50/50**	25/50	40/50	46/50	50/50	49/50	**37/50***	8/50*	29/50*	50/50
Gaussian	49/50	23/50	39/50	48/50*	50/50	49/50	26/50	5/50*	35/50*	50/50
Intermittency	31/50	26/50	39/50	47/50*	50/50	37/50	25/50	5/50*	**37/50***	50/50
Cubic	**50/50**	23/50	42/50	44/50	50/50	**50/50**	31/50*	7/50*	31/50*	50/50

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
* are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

maps, because with the maps, the algorithm had the best search ability in two or more test problems. However, the it could not demonstrate the best global search ability across the test problems and dimensions using any of the chaotic maps. Apart from f_1, f_3, and f_5 (for dimension 30) and f_1 and f_5 (for dimension 50), other maps had better influence on the algorithm in achieving better global search ability than logistic map as shown by the shaded portions in the table.

Table 10 shows the number of function evaluations by the algorithm, using the various chaos maps. When Intermittency map was used and the problem dimension was set to 30, the algorithm had the lowest number of function evaluations in four of the test problems but in three of the test problems under dimension 50, thereby making the algorithm be more robust in terms of search ability than other maps. The shaded portions indicate lower number of function evaluations executed by the algorithm when the corresponding chaotic maps were used compared with logistic map.

Table 11 shows the average ranking of the performance of RIW-PSO when each of the chaotic maps was incorporated into it to solve all the test problems. Each value in the table represents the average rank of the corresponding map in comparison to others across the test problems for each problem dimension. The least value indicates that the associated map performed best, while the largest value indicates that the associated map is the least in performance. Generally, when the problem dimension was set to 30, the algorithm obtained the best convergence precision, demonstrated the best global search ability and stability when logistic map was used. But using Circle and Intermittency maps, less computational effort was needed compared with others. When problem dimension was set to 50, the algorithm obtained the best convergence precision using ICMIC map and it was more stable using Tent map; it did better global search using Piecewise map and required less computational time using Circle map. On the average, Intermittency map performed best when the problem dimension was 30, while Piecewise map performed best when the problem dimension = 50.

6. Conclusions

Chaotic features cause the values of inertia weight to fluctuate between 0 and 1. It affects the velocities and positions of each particle in each iteration to facilitate its local and global search ability as they move to new search regions in the search space.

In this paper, two PSO variants, LDIW-PSO and RIW-PSO algorithms, were implemented with different chaotic maps incorporated into their inertia weight strategies at different times. Their performances were investigated based on the results obtained from numerical simulations, using some well-studied benchmark problems. Mean best solution, standard deviation, success rate, and function evaluations of the algorithms were the instruments of measurement. Results show that, though logistic map could enhance the performance of LDIW-PSO and RIW-PSO, there are other chaotic maps that can make the variants perform better in terms of convergence speed, accuracy, stability, and global search ability.

In terms of average performance, LDIW-PSO performed best using the Intermittency map when the problem dimension is 30. But it performed best using Gaussian map when the problem dimension is 50. On the other hand, RIW-PSO also performed best using the Intermittency map when the problem dimension is 30. But it performed best using Piecewise map when the problem dimension is 50. This is an indication that the Intermittency chaotic map could enhance the performance of the two PSO variants compared with other maps, when the dimensionality of the problem is in the neighbourhood of 30.

However, it should be noted that due to the different search pattern of the chaotic maps, they could be problem dependent. The results presented in this paper can also serve as a platform for informative decision making by practitioners in the process of selecting chaotic maps to be used in the inertia weight formula of LDIW-PSO and RIW-PSO. Further work will be done in applying these variants to real-world problems to test their effectiveness relative to the chaotic maps.

Table 10: Number of function evaluations for test problems using RIW-PSO.

Chaos maps	Dimension = 30					Dimension = 50				
	f_1	f_2	f_3	f_4	f_5	f_1	f_2	f_3	f_4	f_5
None	—	—	$1.945e+05$	—	—	—	—	—	—	—
Logistic	$1.652e+04$	$2.775e+04$	$6.688e+03$	$1.487e+04$	$1.220e+04$	$4.404e+04$	$5.365e+04$	$5.449e+05$	$8.453e+04$	$3.035e+04$
Tent	$1.567e+04^*$	$2.329e+04^*$	$8.775e+03$	$1.415e+04^*$	$1.187e+04^*$	$3.742e+04^*$	$4.385e+04^*$	$3.170e+05^*$	$8.828e+04$	$2.698e+04^*$
Skew Tent	$1.515e+04^*$	$2.896e+04$	$5.458e+03^*$	$1.278e+04^*$	$1.081e+04^*$	$3.785e+04^*$	$4.559e+04^*$	$3.658e+05^*$	$8.329e+04^*$	$2.474e+04^*$
Sine	$1.657e+04$	$2.650e+04^*$	$7.083e+03$	$1.221e+04^*$	$1.218e+04^*$	$4.063e+04^*$	$4.747e+04^*$	$2.541e+05^*$	$6.767e+04^*$	$2.853e+04^*$
ICMIC	$1.629e+04^*$	$2.033e+04^*$	$7.955e+03$	$1.338e+04^*$	$1.117e+04^*$	$3.806e+04^*$	$5.395e+04$	$1.315e+05^*$	$6.426e+04^*$	$2.587e+04^*$
Circle	$1.509e+04^*$	$2.187e+04^*$	$4.986e+03^*$	$1.291e+04^*$	$9.600e+03^*$	$3.682e+04^*$	$4.185e+04^*$	$1.235e+05^*$	$6.011e+04^*$	$2.265e+04^*$
Piecewise	$1.537e+04^*$	$2.708e+04^*$	$8.796e+03$	$1.455e+04^*$	$1.135e+04^*$	$4.032e+04^*$	$\mathbf{3.811e+04}^*$	$1.999e+05^*$	$6.358e+04^*$	$2.615e+04^*$
Gaussian	$\mathbf{1.447e+04}^*$	$2.492e+04^*$	$5.509e+03^*$	$1.243e+04^*$	$9.888e+03^*$	$\mathbf{3.583e+04}^*$	$4.649e+04^*$	$2.899e+05^*$	$5.062e+04^*$	$2.292e+04^*$
Intermittency	$1.976e+04$	$\mathbf{1.759e+04}^*$	$\mathbf{3.740e+03}^*$	$\mathbf{1.026e+04}^*$	$\mathbf{7.970e+03}^*$	$4.391e+04^*$	$3.978e+04^*$	$\mathbf{1.112e+05}^*$	$\mathbf{4.000e+04}^*$	$\mathbf{1.874e+04}^*$
Cubic	$1.630e+04^*$	$3.213e+04$	$7.205e+03$	$1.421e+04^*$	$1.222e+04$	$4.114e+04^*$	$4.898e+04^*$	$2.153e+05^*$	$6.709e+04^*$	$2.884e+04^*$

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.
* are the better results obtained by the algorithms relative to the corresponding chaotic maps, compared to Logistic map.

Table 11: Average ranking of the performance of RIW-PSO relative to the various chaotic maps.

Chaos maps	Problem dimension = 30					Problem dimension = 50				
	Mean fitness	Standard deviation	Success rate	Function evaluation	Average performance	Mean fitness	Standard deviation	Success rate	Function evaluation	Average performance
None	11	10.8	5.8	11	9.65	11	10.8	6.2	11	9.75
Logistic	**4.2**	**4.6**	**2**	8	4.70	6.2	6.2	3.8	9.6	6.45
Tent	5.8	5.8	2.8	6.4	5.20	5.2	**3.4**	3.8	6.4	4.70
Skew Tent	6	5.2	2.8	4.6	4.65	6.6	3.6	4.4	6	5.15
Sine	5.6	5	2.4	6.2	4.80	4.8	6.8	2.8	7	5.35
ICMIC	6	5.6	2.4	5.4	4.85	**4**	6.2	2.8	5.6	4.65
Circle	6	5.4	3.6	**2.8**	4.45	5.8	6.4	2.4	**2.4**	4.25
Piecewise	5	5	2.8	7.2	5.00	4.6	4.4	**2.2**	4.2	**3.85**
Gaussian	5.6	5.8	3	3.2	4.40	5.4	5.2	3.2	3.8	4.40
Intermittency	4.8	6	3.2	**2.8**	**4.20**	6.2	6.2	3.4	2.8	4.65
Cubic	6	6.8	3	8.4	6.05	6.2	6.8	2.4	7.2	5.65

The values in bold are the best results obtained by the algorithms using the corresponding chaotic map incorporated into it compared with others.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, October 1995.

[2] M. S. Tavazoei and M. Haeri, "Comparison of different one-dimensional maps as chaotic search pattern in chaos optimization algorithms," *Applied Mathematics and Computation*, vol. 187, no. 2, pp. 1076–1085, 2007.

[3] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[4] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[5] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 94–100, May 2001.

[6] W. Wang and L. Qiu, "Optimal reservoir operation using PSO with adaptive random inertia weight," in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI '10)*, vol. 3, pp. 377–381, Sanya, China, October 2010.

[7] Y.-L. Gao and Y.-H. Duan, "A new particle swarm optimization algorithm with random inertia weight and evolution strategy," in *Proceedings of the International Conference on Computational Intelligence and Security Workshops (CISW '07)*, pp. 199–203, December 2007.

[8] M. Pant, T. Radha, and V. P. Singh, "Particle swarm optimization using Gaussian inertia weight," in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA '07)*, vol. 1, pp. 97–102, Sivakasi, India, December 2007.

[9] L.-Y. Chuang, C.-H. Yang, and J.-C. Li, "Chaotic maps based on binary particle swarm optimization for feature selection," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 239–248, 2011.

[10] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, p. 475, Kumamoto, Japan, September 2007.

[11] L. D. S. Coelho, "A quantum particle swarm optimizer with chaotic mutation operator," *Chaos, Solitons & Fractals*, vol. 37, no. 5, pp. 1409–1418, 2008.

[12] A. H. Gandomi, G. J. Yun, X.-S. Yang, and S. Talatahari, "Chaos-enhanced accelerated particle swarm optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 18, no. 2, pp. 327–340, 2013.

[13] Y. Gao, X. An, and J. Liu, "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation," in *Proceedings of the IEEE International Conference on Computational Intelligence and Security (CIS '08)*, vol. 1, pp. 61–65, Suzhou, China, December 2008.

[14] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons & Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[15] Y. H. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 1945–1950, Washington, DC, USA, 1999.

[16] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds., vol. 1447, pp. 591–600, Springer, Berlin, Germany, 1998.

[17] G. Chen, X. Huang, J. Jia, and Z. Min, "Natural exponential inertia weight strategy in particle swarm optimization," in *Proceedings of the 6th World Congress on Intelligent Control and Automation (WCICA '06)*, pp. 3672–3675, Dalian, China, June 2006.

[18] R. F. Malik, T. A. Rahman, S. Z. M. Hashim, and R. Ngah, "New particle swarm optimizer with sigmoid increasing inertia weight," *International Journal of Computer Science and Security*, vol. 1, no. 2, pp. 35–44, 2007.

[19] X. Shen, Z. Chi, J. Yang, and C. Chen, "Particle swarm optimization with dynamic adaptive inertia weight," in *Proceedings of the IEEE International Conference on Challenges in Environmental Science and Computer Engineering (CESCE '10)*, vol. 1, pp. 287–290, Wuhan, China, March 2010.

[20] K. Kentzoglanakis and M. Poole, "Particle swarm optimization with an oscillating inertia weight," in *Proceedings of the 11th Annual Genetic and Evolutionary Computation Conference (GECCO '09)*, pp. 1749–1750, July 2009.

[21] H.-R. Li and Y.-L. Gao, "Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation," in *Proceedings of the 2nd International Conference on Information and Computing Science (ICIC '09)*, vol. 1, pp. 66–69, Manchester ,UK, May 2009.

[22] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: an overview," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[23] J. Xin, G. Chen, and Y. Hai, "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," in *Proceedings of the International Joint Conference on Computational Sciences and Optimization (CSO '09)*, pp. 505–508, Sanya, China, April 2009.

[24] Y. H. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 101–106, Seoul, Republic of korea, May 2001.

[25] G. I. Evers, *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization [M.S. thesis]*, University of Texas-Pan American, Edinburg, Tex, USA, 2009.

[26] A. M. Arasomwan and A. O. Adewumi, "An adaptive velocity particle swarm optimization for high-dimensional function optimization," in *Proceedings of the IEEE Congress Evolutionary Computation (CEC '13)*, pp. 2352–2359, 2013.

[27] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems," *Journal of Global Optimization*, vol. 31, no. 4, pp. 635–672, 2005.

[28] S. Chetty and A. O. Adewumi, "Three new stochastic local search algorithms for continuous optimization problems," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 675–721, 2013.

[29] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained

global optimization," *Optimization Methods and Software*, vol. 26, no. 6, pp. 945–970, 2011.

[30] R. Akbari and K. Ziarati, "A rank based particle swarm optimization algorithm with dynamic adaptation," *Journal of Computational and Applied Mathematics*, vol. 235, no. 8, pp. 2694–2714, 2011.

[31] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 120–127, Honolulu, Hawaii, USA, April 2007.

[32] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.

[33] M. E. H. Pedersen, "Good parameters for particle swarm optimization," Tech. Rep. HL1001, Hvass Laboratories, 2010.

[34] M. A. Arasomwan and A. O. Adewumi, "On the performance of linear decreasing inertia weight particle swarm optimization for global optimization," *The Scientific World Journal*, vol. 2013, Article ID 860289, 12 pages, 2013.

[35] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "Benchmarking projection-based real coded genetic algorithm on BBOB-2013 noiseless function testbed," in *Proceeding of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*, pp. 1193–1200, ACM, New York, NY, USA, 2013.

Improved Particle Swarm Optimization for Global Optimization with(out) some Control Parameters

Akugbe Martins ARASOMWAN and Aderemi Oluyinka ADEWUMI[1]

School of Mathematics, Statistics & Computer Science
University of KwaZulu-Natal,
Private Bag X54001,
Durban 4000, South Africa

Abstract

This paper achieved two major goals. First, it experimentally showed that the Basic Particle Swarm Optimization (BPSO) technique can perform efficiently without using some (or any) of the control parameters in the particle velocity update formula. Second, the problem of premature convergence associated with PSO technique when optimizing high dimensional multi-modal optimization problems was ameliorated. In achieving these goals, some modifications were done to BPSO. Some of the modifications involved making the velocity limits of the particles to decrease dynamically depending on the progressive minimum and maximum dimensional values of the entire swarm. The decreasing nature of the velocity limits was used to control the exploration and exploitation activities of the modified BPSO (*M*-BPSO). Different experiments were carried out to ascertain the possibilities of implementing BPSO and *M*-BPSO without some and all of the control parameters in the particle's updating formula. Typical global optimization benchmark problems were used to validate the proposed modifications through empirical studies and results of *M*-BPSO were compared with BPSO. The results of some of the variants of *M*-BPSO were also compared with those of two other efficient optimization algorithms in literature. All the experimental results show that the proposed *M*-BPSO algorithm is very effective and was found superior in

[1] Corresponding Author: Email: adewumia@ukzn.ac.za

performance to the other algorithms adopted for comparison in terms of solution quality, convergence speed, global-local search ability, and stability.

Keywords: Particle Swarm Optimization, Global Optimization, Parameter Control, Continuous Problems, Inertia weight

1. Introduction

The increase in computationally complex problems found in engineering, sciences, etc., is a source of continual motivations to researchers in the field of optimization. The outcome of such motivations is the development of efficient algorithms inspired by nature. Biologically inspired algorithms have proven to be efficient in handling computationally complex problems with competence and many successes have been recorded. Swarm intelligence, a class of biologically inspired algorithms, comprises algorithms that are population-based which do not depend on the gradient, continuity or differentiability for problem being optimized or solved. All that is required is the computability of the problem. Some of the algorithms which belong to the class of swarm intelligence are Ant Colony Optimization (ACO), Bees Algorithm (BA), Firefly Algorithm (FA) as well as Particle Swarm Optimization (PSO).

Apart from being population-based, PSO is a stochastic and adaptive optimization technique which was inspired by the social interaction in human beings and animals like bird flocking or fish schooling. It was introduced by Kennedy and Eberhart in 1995 [7, 15]. To optimize a problem, the basic PSO (BPSO) technique is initialized by randomly distributing a set of particles (potential solutions) in a solution search space. The particles are made to fly through the problem space and allowed to repeatedly search for optimal solution over a period of time. In the process, other particles follow the current optimum particle while their positions

and velocities are updated in each iteration relative to the personal experience of each particle as well as that of the entire particles. The quality of the solution found by each particle is obtained using the objective function of the problem being optimized. Represented in equations (1) and (2) are the respective updating formulas for the velocity and position of each particle. In a physical *d*-dimensional search space, the particle's position is represented as $\vec{X}_i = (x_{i1}, \ldots, x_{id})$ while its velocity is represented as $\vec{V}_i = (v_{i1}, \ldots, v_{id})$.

$$\vec{V}_i^{t+1} = \underbrace{\omega\vec{V}_i^t}_{\text{Inertia component}} + \underbrace{c_1\vec{r}_1\left(\overrightarrow{pb}_i^t - \vec{X}_i^t\right)}_{\text{Cognitive component}} + \underbrace{c_2\vec{r}_2\left(\overrightarrow{gb}^t - \vec{X}_i^t\right)}_{\text{Social component}} \quad (1)$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1} \quad (2)$$

From equation (1), it is clear that the velocity updating formula is made up of three different components. These components are briefly described as follows:

(i) *Inertia component:* This component provides the necessary momentum for particles to roam across the search space. It is made up of the inertia weight parameter (ω) which help to balance the exploration and exploitation activities of the algorithm; and the particle's previous velocity at t^{th} iteration ($\vec{V}_i^t$). In the original PSO technique, the value of 1 was used for ω. This component models the tendency of a particle to remain in the same direction it has been navigating.

(ii) *Cognitive component:* This component represents the memory or personal thinking of the particle. It consists of a constant value c_1 representing the self-confidence of a particle, a vector of random numbers ($\vec{r}_1$) uniformly generated in the interval [0,1] and the distance between the best position the particle has ever visited known as personal best ($\overrightarrow{pb}_i^t$) and its current position ($\vec{X}_i^t$). This component models the linear attraction of the particle towards its own best experience.

(iii) *Social component:* This component represents the knowledge and collaborative effect of the particles of the swarm, in finding the global optimal solution. It is made up of a constant value c_2 representing the swarm confidence, another vector of random numbers ($\vec{r}_2$) uniformly generated in the interval [0,1] and the distance between the swarm's best position known as global best ($\overrightarrow{gb^t}$) at t^{th} iteration and the particle's current position ($\vec{X}_i^t$). This component models the linear attraction of the particle towards the best experience of the swarm.

During execution of the BPSO technique, there are possibilities that the design variables can go outside their lower ($X_{\min}$) and upper ($X_{\max}$) boundaries and take values which could lead to divergence. In such situations, the common practice is to artificially bring the affected particle back to the search space boundary. In the same vain, velocities of the particles are clamped within some specified maximum velocity bounds [$V_{\min}$, $V_{\max}$], where $V_{\min}$ is the velocity lower bound and $V_{\max}$ is the velocity upper bound. This is because the velocity updating formula is stochastic and the velocity may become too high which could lead the particles becoming uncontrolled and exceed the search space.

```
If x_i < x_min
   x_i ← x_min
else if x_i > x_max
   x_i ← x_max
end if
```

Algorithm 1

```
If v_i < v_min
   v_i ← v_min
else if v_i > v_max
   v_i ← v_max
end if
```

Algorithm 2

From the foregoing, two major observations can be made:

(i) The BPSO consists of 3 major steps in sequential order of generating positions and velocities for all the particles that make up the swarm; updating the velocities of the particles; and updating the positions of the particles.

(ii) The BPSO depends on some parameters which control its operations and efficiency namely: Inertial weight parameter (ω); Self-confidence of particle (c_1) and swarm

confidence (c_2), which are also known as acceleration constants (or coefficients); Cognitive and social random factors $\vec{r}_1$ and $\vec{r}_2$; Particles' velocity clamping, [$V_{\min}$, $V_{\max}$]; and Particles' position clamping, [$X_{\min}$, $X_{\max}$].

Although, the topology of particle's neighbourhood may also influence the trajectories of particles but this is not considered in this paper.

Naturally, BPSO technique combines local search method (through self-experience) with global search methods (through neighbouring experience), attempting to balance exploration and exploitation. Exploration is the ability the algorithm to explore new regions of the search space, while exploitation is the ability to search a smaller region more thoroughly. It is widely accepted that BPSO technique has good global search ability but weak local search ability, because it can easily locate the good area of the solution space in which good solutions are located but finding the best solution proves difficult. This challenge has motivated many researchers to introduce many new BPSO variants while others tried to improve on existing variants. [1, 2, 11, 18]. This paper has made some improvements on the BPSO technique to make it simpler but more effective. It has been empirically shown that the algorithm can perform efficiently without some of the control parameters listed above. This was done by using some well-known benchmark problems extensively used in the literature for the evaluations of metaheuristics to validate the proposed modified BPSO (*M*-BPSO). The importance of BPSO to enhance its local search ability for computational effectiveness and efficiency is one of the things achieved by *M*-BPSO.

The remaining part of the paper is organized as follows. In section 2, the control parameters in BPSO technique are reviewed. The proposed modification to BPSO algorithm is described in section 3. Sections 4 gives the methodological approach used in carrying out the numerical

simulations in the paper and Section 5 reports and discusses the numerical simulations and results. Finally, Section 6 concludes the paper.

2. Control parameters in particle swarm optimization technique

When PSO was proposed in 1995, a new world was opened to researchers in the field of optimization. Since then quite a number of researches have been done, with the singular focus to improve on the performance and robustness of the technique in handling optimization problems. These researches have been in terms of introducing velocity limit, V_{max} [7] and inertia weight, ω [23] into PSO and their various improvements [1, 3]; introduction of constriction factor [5], creating other variants of PSO [11] and hybridization of PSO with other algorithms [13]. Briefly reviewed below are the control parameters mentioned earlier.

Inertia weight and its variants:

This parameter, commonly represented as ω, was introduced into PSO by [23]. The inspiration behind its introduction was the desire to balance the scope of local and global searches and reduce the importance of velocity clamping during the optimization process. Over the years several inertia weight strategies have been proposed to dynamically adjust the value of ω in each iteration [17 19, 20]. These strategies include random [8], chaotic random [10], linear decreasing [24, 28], and chaotic linear decreasing [10]. In [24, 28], the linear decreasing inertia weight strategy decreases from a value of 0.9 to 0.4, however there are cases where values other than 0.9 or 0.4 are used [16]. Though it enhanced the performance of PSO, it usually got into local optimum when solving functions with more apices [10]. In [8] it was experimentally found that random inertia weight strategy increases the convergence in PSO and could find good results with most functions. A chaotic term was included to the random as well as the linear decreasing inertia weight strategies in [10]. These strategies were experimentally proved to be superior to the random and linear decreasing strategies in terms of convergence speed, global search ability and convergence precision. Other inertia weight strategies include Fuzzy adaptive inertia weight which is dynamically adjusted on the basis of

fuzzy sets and rules in each iteration [25] and adaptive inertia weights which are dynamically adjusted based on some feedback parameters like swarm particle fitness, particle rank, distance to particle, global best positions, and particle success rate [20]. All these inertia weight strategies have been experimentally proved to enhance the performance of PSO with varying degree of successes.

Maximum velocity:

The particle velocity based on equation (1), without restriction, can grow unbounded while the particle oscillates around an optimum, increasing its distance to the optimum on each iteration. This initiated the introduction of velocity clamping effect to avoid the phenomenon of "swarm explosion". This idea was introduced by Eberhart and Kennedy in 1995 [9]. It improves the performance of PSO because it helps particles take reasonably sized steps so as to rake through the search space rather than bouncing about excessively. Efforts have been made in time past to eliminate the use of V_{max} but researches have shown that velocity clamping has become a standard feature of PSO [9].

Wrong setting of maximum velocity bounds for particles could have negative affect on the performance of PSO algorithms because it may either make the particles do too much exploration or exploitation (if the value is too high or too low). Different efforts have been made by researchers to determine appropriate values for the velocity limits of particles in order to improve on the performance of PSO [1, 9, 24]. The three major methods that appear in literature, for computing the velocity clamping (V_{min} and V_{max}) are recorded in [1]:

(i) multiplying the search space range with certain percentage (δ), i.e. $V_{max} = \delta(X_{max} - X_{min})$ and $V_{min} = -V_{max}$.

(ii) multiplying both the minimum and maximum limits of the search space separately with certain percentage (δ), i.e. $V_{max} = \delta(X_{max})$ and $V_{min} = \delta(X_{min})$.

(iii) assigning the search space upper limit to V_{max}, i.e., $V_{max} = X_{max}$

Some of the different values used by different authors for $\delta \in (0,1]$ to determine velocity clamping for particles are shown Table 1.

Acceleration coefficients

These parameters are commonly represented as c_1 (cognitive scaling parameter) and c_2 (social scaling parameter) and are positive values. The values of 2.0 as originally assigned to these parameters when PSO was introduced in [13], have been adopted by many researchers over the years [1, 4, 10, 11, 29]. As a result of the sensitive roles of these parameters in the performance of PSO, other researchers have attempted to adjust them through empirical studies. Such researches include [30 – 33]. In [33], the role acceleration coefficients play in the performance of PSO was investigated by using unsymmetrical transfer range of acceleration coefficients. The simulations that were carried out showed an improved optimum solution for most of the benchmarks that were used was observed when changing c_1 from 2.75 to 1.25 and changing c_2 from 0.5 to 2.25, over the full range of the search.

Table 1: Various values for δ in the literature

δ	Velocity clamping formula	Reference
0.2	$V_{max} = \delta * X_{max}$	[6]
0.05	$V_{max} = \delta * X_{max}$ $V_{min} = \delta * X_{min}$	[1, 22]
0.15	$V_{max} = \delta * (X_{max} - X_{min})$	[1, 9, 29]
0.5	$V_{max} = \delta * (X_{max} - X_{min})$	[1, 3]

In [31], New PSO (NPSO) was proposed. In it the cognitive acceleration coefficient c_1 was split into good experience component c_{1g} and bad experience component c_{1b} to help the particles move towards their previous best positions and away from their previous worst positions in order to facilitate exploration capability. For the purpose of improvement Anti-Predatory PSO (APSO) was proposed by [32], where the cognitive acceleration coefficients c_1 was split into good experience component c_{1g} and bad experience component c_{1b} and c_2 was also split was split into good experience component c_{2g} and bad experience component c_{2b}. The bad experiences help particles to by-pass their previous worst positions while good

experiences help particles to move towards their previous best positions. Similarly in [30], Time-Varying Acceleration Coefficients PSO (PSO-TVAC) was introduced to enhance the global search in the early part of the optimization and to encourage the particles to converge toward the global optimum at the end of the search. This was achieved by linearly decreasing the cognitive parameter c_1 from a high value c_{1max} to a low value c_{1min} but the social parameter c_2 linearly increased from a low value c_{2min} to a high value of c_{2max}.

Relationships among the control parameters in the velocity update formula

The importance of parameter selection in PSO algorithm has drawn attention from many researchers. However, the general belief in PSO community has been that the inertia weight balances exploration and exploitation activities in PSO algorithm. Researches have shown that that inertia weight cannot balance exploration and exploitation by itself in PSO algorithm but in cooperation with some other (control) parameters [12, 24]. Different researchers have proposed and used different sets of control parameter values which are presented in Table 2.

Table 2: Various BPSO values for inertia weight and acceleration constants parameters in the literature

Inertia weight (ω)	Cognitive component acceleration constant (c_1)	Social component acceleration constant (c_2)	Reference
0.729	1.494	1.494	[5]
0.6	1.7	1.7	[27]
0.729	2.041	0.948	[12]
0.715	1.7	1.7	[12]
0.72	1.49	1.49	[18]
0.6	1.8	1.8	[14, 21]
Computed using formula	2.8	1.3	[3]

3. The proposed modifications in PSO technique

The efficient optimizing power of PSO lies in the balancing of exploration and exploitation activities. As earlier established, the inertia weight, acceleration constants, random factors and velocity threshold play important roles in the exploration and exploitation ability of PSO algorithm, though their selections could be problem-dependent, laborious and time

consuming except the random factors. The following observation motivated and facilitated the proposed modification done to the basic PSO in this paper.

(i) Experimental studies in [24] show that large V_{max} enhances exploration while small V_{max} enhances exploitation. The implication of this is that, if V_{max} can be dynamically varied from some large value to some small value, then it can solely play the role of the inertia weight parameter. Besides, clamping the velocity of a particle can change the step size and direction of the particle. As each dimension is optimized independently, the particle moves toward the global best on each dimension with a speed depending on the velocity limits, thereby creating opportunities for particles to comb the search space a bit more thoroughly than when their velocities are unclamped [9]. Above all, exploration and exploitation in PSO could better be addressed by working directly with the velocities of the particles because it is the direct determinant of the particles' step sizes.

(ii) Without doubt, the appropriate selection of c_1 and c_2 can accelerate convergence and avoid being trapped in local optimums. Combining them with the random factors (r_1 and r_2) to weight the cognitive and social components as shown in equation (1) make particles to base their searches in the interval [0,2] centred on particle's personal best and swarm global best. This could lead a problem of particles jumping over the optimal solution if large weighting factors are generated or the number of iterations to locate the optimal solution may be increased if small weighting factors are generated [2, 15]. Some efforts could be saved if c_1 and c_2 are eliminated (i.e., they are allowed to take the value of 1).

(iii) The purpose why r_1 and r_2 was included into PSO algorithm was to make it stochastic to facilitate exploration. Despite the good roles they play, it is possible that PSO can still perform well if they are not included in the velocity formula. To compensate for this exploration could be initiated in some other part (e.g., position clamping, etc.) of the algorithm.

Based on these observations, the modifications were done to the basic PSO algorithms are now described as follows:

(i) During each iteration, the largest dimension value (L_d) and the smallest dimension value (S_d) among the dimensions of all the particles, were obtained according to equations (3) and (4), where, x_i^j is the i^{th} particle with j^{th} dimension.

$$L_d \leftarrow \max_i \left(\max_j \left(x_i^j \right) \right) \tag{3}$$

$$S_d \leftarrow \min_i \left(\min_j \left(x_i^j \right) \right) \tag{4}$$

(ii) The upper limit x_{max} and lower limit x_{min} of the solution search space for the particles were obtained according to equations (5) and (6), where | . | means absolute value.

$$x_{max} \leftarrow max(|L_d|, |S_d|) \tag{5}$$

$$x_{min} \leftarrow -x_{max} \tag{6}$$

(iii) After obtaining x_{max} and x_{min}, they are used to compute the upper (v_{max}) and lower (v_{min}) particle velocity limits as defined in equations (7) and (8).

$$v_{max} \leftarrow \mu x_{max} \tag{7}$$

$$v_{min} \leftarrow \mu x_{min} \tag{8}$$

where, μ is a velocity clamping percentage. It serves as a scaling factor of the upper and lower solution space limits to help reduce the velocity range for particles. As the algorithm progresses, the velocity range of the particles decreases, thereby reducing the distance each particle should exploit for better solution and the smaller the velocity range the higher the exploitation by the particles.

(iv) Next, an integer random number p is generated in the interval [1, S] where S is the swarm size and out of the swarm, p number of particles are randomly selected

(v) A value h was obtained by dividing the problem dimension by 2, after which h number of dimensions were randomly selected for each particle picked in (iv). The position and velocity for each of the dimension are uniformly re-initialized based on the new x_{min}, x_{max}, v_{min}, and v_{max} obtained from equations (5) – (8). This process is represented by equations (9) – (11).

$$h \leftarrow \left(\frac{Problem\ dimension}{2}\right) \tag{9}$$

$$x_i^j \leftarrow (x_{max} - x_{min}) \times rand[0,1] + x_{min} \tag{10}$$

$$v_i^j \leftarrow (v_{max} - v_{min}) \times rand[0,1] + v_{min} \tag{11}$$

where i and j are the index and dimension respectively, of a randomly selected particle in (iv). This method help the algorithm achieve some level of exploration by providing it with the particles opportunities of leaving their current positions to other parts of the search space, thus helping to escape getting stuck in local optimum. This happens throughout the process of the algorithm.

(vi) When the particles' positions are being updated, contrary to the common method (Algorithm 1) of forcing the particles that obtain values outside the search space to the search boundaries, they are adjusted using Algorithm 3. This method can also enable the algorithm jump out of local optimum and does some exploration to search other parts of the search space. This also happens throughout the process of the algorithm.

```
If x_i < x_min
  x_i ← x_min + (x_min − x_i) *
random(0,1)
else if x_i > x_max
  x_i ← x_max − (x_i − x_max) *
random(0,1)
end if
```

Algorithm 3

The algorithm for the proposed modified basic PSO (*M*-BPSO) is presented in Algorithm 4. The shaded portions show the modifications that have been incorporated into BPSO.

```
Begin M-BPSO Algorithm
  Input:      f: the function to optimize
              s: the swarm size
              d: the problem dimension
              X_min, X_max : decision variable search range
              V_min, V_max : particle velocity limits
  Output:     x*: the best particle position found (global
  best)
              f*: the best fitness value found
  Initialize: position x_i = (x_i1, …, x_id) and velocity v_i = (v_i1,
                …, v_id), for all particles in problem space
  evaluate f(x_i) in d variables and get pbest_i, (i = 1, …, s)
  gbest ← best of pbest_i
  While stopping criteria is false do
    if it is necessary, compute inertia weight (ω) if it is not
    a constant
    compute new X_min, X_max using equations (3) – (6)
    compute new V_min, V_max using equations (7) and (8)
    generate an integer random value p ∈ U[1,S] and randomly
      pick p particles from the swarm
    compute h using Equation (9)
    using Equations (10) and (11) randomly re-initialize the
    positions and corresponding velocities of h randomly
    selected dimensions of the p particles, based on the new
    X_min, X_max, V_min and V_max
    randomly reinitialize velocities for particles using the
           new vr
    Repeat for s times
     Repeat for d times
        update v_i for particle using equation (1)
        validate for velocity boundaries using Algorithm 2
        update x_i for particle using equation (2)
        validate for position boundaries using Algorithm 3
        compute f(x_i)
     End Repeat for d
     compute f(x_i)
     obtain new pbest_i
     If f(x_i) < f(pbest_i) then pbest_i ← x_i
     If f(x_i) < f(gbest) then
       gbest ← x_i
       f(gbest) ← f(x_i)
     end if
    End Repeat for s
  End while
  x* ← gbest
  f* ← f(gbest)
  Return x* and f*
End M-BPSO Algorithm
```

Algorithm 4

4. Methodology

The methods enumerated below were applied to systematically achieve the set goals in this paper.

(i) Obtain good values from the literature for the parameters (ω, c_1, c_2, V_{min} and V_{max}) to implement BPSO .

(ii) Create different variants for BPSO and *M*-BPSO as shown Table 3.

Table 3: Different variants for BPSO and *M*-BPSO

Variants of BPSO	Variants of *M*-BPSO	Variants were implemented with these parameters in velocity updating formula
$BPSO_1$	*M*-$BPSO_1$	ω, c_1, c_2, r_1 and r_2 (i.e. all parameters were used)
$BPSO_2$	*M*-$BPSO_2$	c_1, c_2, r_1 and r_2 (i.e. inertia weight parameters was not used)
$BPSO_3$	*M*-$BPSO_3$	ω, c_1 and c_2 (i.e. random factors parameters were not used)
$BPSO_4$	*M*-$BPSO_4$	ω, r_1 and r_2 (i.e. acceleration constants parameters were not used)
$BPSO_5$	*M*-$BPSO_5$	r_1 and r_2 (i.e. inertia weight& acceleration constants parameters were not used)
$BPSO_6$	*M*-$BPSO_6$	c_1 and c_2(i.e. inertia weight& random factor parameters were not used)
$BPSO_7$	*M*-$BPSO_7$	None of the parameters (i.e. no parameters were used)

(iii) Implement the various variants in Table 3 using some selected well-studied benchmark continuous optimization problems in ithe literature and compare results between the respective variants of BPSO and *M*-BPSO. This stage will ascertain the possibilities of implementing BPSO and *M*-PSO without some (or all) of the control parameters as well as their corresponding performances

(iv) If any of the *M*-BPSO variants perform better, select among them those with few number of parameters and validate their performance against the following algorithms – modified attractive repulsive PSO (MARPSO) [11] and another swarm intelligence technique, Bioluminescent swarm optimization algorithm (BSO) [21].

(v) Measure the performances of all the algorithms using the following criteria [1,34,35]:

a. Best fitness solution: This is the best fitness solution among all the best fitness solutions obtained by an algorithm in all the specified independent runs

b. Mean best fitness solution: This is the average of all the best fitness solutions. It is a measure of the precision (quality) of the result that the algorithm can get within given iterations in all the specified independent runs

c. Standard deviation (Std. Dev.) of mean best fitness solution over 50 runs: This measures the algorithm's stability and robustness

d. Average number of iteration an algorithm met the specified success criteria. This was mainly used to judge the performance of the variants when there is a tie in success rate (SR) between the respective competing variants

e. Success rate (SR) $= \frac{Number\ of\ successful\ runs}{Total\ number\ of\ runs} \times 100$: This is the rate at which the success criteria is met during the independent number of runs and is a reflection of the global search ability and robustness of the algorithm

5. Numerical simulations

Two sets of experiments were conducted in this study. In the first set, all the variants of *M*-BPSO and BPSO as defined in Table 3, was implemented using the same testing conditions and their performances were compared. The essence of this set of experiments is to test the effects of the various control parameters in the particle's velocity updating formula in Equation (1) on the respective variants. In the second sets of experiments, some of the variants of *M*-BPSO that that are independent on the inertia weight parameter (ω) were selected and tested against two existing optimization algorithms that are very efficient, Modified Attractive-Repulsive PSO (MARPSO) in [11] and Bioluminescent Swarm Optimization (BSO) algorithm in [21]. The essence of this set of experiments was to test if the proposed algorithm (*M*-BPSO) could favourably compete with existing optimization algorithms. The application software was developed in Microsoft Visual C# programming language.

5.1. Test problems

For the first set of experiments, a total of 6 scalable test problems in Table 4 and 4 non-scalable problems in Table 5 were used. These problems adapted from [1, 14, 20], have

diverse complexities and multimodality common among many complex global optimization problems. They have different characteristics (US – unimodal separable, UN – unimodal non-separable, MS – multimodal separable, MN – multimodal non-separable). For the second set of experiments, the test problems and other testing conditions were used as recorded in [11, 21].

5.2. Parameter setting

For the first set of experiments, the maximum number of iterations allowed was 3000. The dimension for the problems in Table 4 was 30 while in Table 5 the dimensions of *Easom*, *Schaffer's f6* and *Shubert* were each set to 2, but that of *Salomon* was 5. A swarm size of 20 was used in all the experiments and 50 independent runs were conducted to collect data for analysis. The termination criteria for all the algorithms were set to be the maximum number of iterations relative. A run by any of the algorithms was recorded successful when the mean best fitness value at the end of the maximum iteration was less than -0.999999 (for *Easom*), -186.7308 (for *Shubert*) and less than 10^{-5} for other problems. The velocity clamping percentage parameter (δ) was 0.25 for *M*-BPSO; ω = 0.715 and $c_1 = c_2 = 1.7$ for both *M*-BPSO and BPSO based on the findings in the experimental studies in [12]; velocity thresholds ($V_{\min}$ and $V_{\max}$) were dynamically obtained using equations (7) and (8) for *M*-BPSO; while for BPSO, $V_{\max} = 0.05 \times X_{\max}$, $V_{\min} = 0.05 \times X_{\min}$ based on the findings in some of our experiments that they make BPSO perform efficiently, where $X_{\min}$ and $X_{\max}$ are the fixed minimum and maximum values of the domain for the decision variables. The parameters r_1 and r_2 were randomly generated using the uniform random number generator.

Table 4: Scalable Benchmark problems

No.	Problem	Formulation	Feature	Search range
1	Ackley	$f(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	MN	±32
2	Griewank	$f(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	MN	±600

3	Levy	$f(\vec{x}) = sin^2(\pi y_1) + \sum_{i=1}^{d-1}(\mathrm{y_i} - 1)^2(1 + 10\,sin^2(\pi y_i + 1))$ $+(y_d - 1)^2(1 + sin^2(2\pi x_d))$, where $\mathrm{y_i} = 1 + \frac{\mathrm{x_i} - 1}{4}$, and $i = 1,2, \ldots, \mathrm{d}$	MN	±10
4	Noisy Quadric	$f(\vec{x}) = \sum_{i=1}^{d} ix_i^4 + random(0,1)$	US	±1.28
5	Rastrigin	$f(\vec{x}) = \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	MS	±5.12
6	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{d-1}\left(100\left(x_{i+1} - x_i^2\right)^2\right) + (x_i - 1)^2$	UN	±30

Table 5: Nonscalable Benchmark problems

No.	Problem	Formulation	Feature	Search range
1	Easom	$f(\vec{x}) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$	UN	±100
2	Salomon	$f(\vec{x}) = -cos\left(2\pi\sum_{i=1}^{d} x_i^2\right) + 0.1\sqrt{\sum_{i=1}^{d} x_i^2} + 1$	MN	±100
3	Schaffer's f6	$f(\vec{x}) = 0.5 + \frac{sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{(0.001(x_{i+1}^2 + x_i^2) + 1)^2}$	MN	±100
4	Shubert	$f(\vec{x}) = \prod_{i=1}^{d}\left(\sum_{j=1}^{5} jcos((j+1)x_i + j)\right)$	MN	±10

5.3. Experimental results and discussions

Results obtained from all the experiments are presented in Tables (6) – (24) and discussed in this sub-section to show the overall computational effectiveness and efficiencies of all the algorithms that were compared in the paper. The measurement criteria stated in Section 4 were used. In all the tables "*" means no result was computed because the algorithm could not meet the success criteria in all the runs for the particular test problem. Bold values indicate best results obtained among the competing algorithms.

5.3.1. Results for the non-scalable test problems

Tables (6) – (12) show the results obtained by the respective variants of BPSO and *M*-BPSO for the low-scaled benchmark problems. The results reflect their performances when all, some and none of the control parameters were used in the particles' velocity update formula.

For *Schaffer's f6* and *Salomon* problems, all the variants of the proposed *M*-PSO not only outperformed those of BPSO, but were able to obtain the global minimum and 100% success rate for the problems. The consistencies in their performances and the successfulness of *M*-$BPSO_7$ show that the control parameters have no significant effects on the computational effectiveness and efficiency of the proposed algorithm. For *Easom* problem, four variants of *M*-BPSO (i.e., *M*-$BPSO_3$, *M*-$BPSO_5$, *M*-$BPSO_6$ and *M*-$BPSO_7$) performed better than their respective counterpart of BPSO variants. On the other hand three of the variants of BPSO (i.e., $BPSO_1$, $BPSO_2$ and $BPSO_4$) performed better than their respective counterpart of *M*-PSO variants. From the results obtained as Best fitness, Mean fitness and standard deviation by *M*-BPSO variants in terms of magnitude, it was observed that the proposed algorithm was consistent in its performance irrespective of the presence of all, some or none of the control parameters in the velocity formula. For *Salomon* problem, four variants of *M*-BPSO (i.e., *M*-$BPSO_1$, *M*-$BPSO_3$, *M*-$BPSO_6$ and *M*-$BPSO_7$) performed better than their respective counterpart of BPSO variants, but were unable to meet the success criteria like BPSO variants. On the other hand three of the variants of BPSO (i.e., $BPSO_2$, $BPSO_4$ and $BPSO_5$) performed better than their respective counterpart of *M*-PSO variants. Another interesting observation that was made is that, *M*-BPSO variants were consistent in their performances in terms of magnitude, irrespective of the presence of any or none of the control parameters in the velocity formula.

The inconsistencies of the performances of BPSO variants in all the results presented in the tables show that the presence of some or none of the control parameters in the particle velocity update formula have some significant effects on it. Surprisingly, $BPSO_6$ obtained better results than $BPSO_1$ in *Schaffer's f6*, $BPSO_4$ obtained better results than $BPSO_1$ in *Shubert*, and $BPSO_4$ had equal performance with $BPSO_1$ in *Easom* problems. All these are clear indications that there are possibilities for BPSO algorithm to perform better when some

of the control parameters are not used in the particle's velocity update formula than when all the parameters are used. These show that the insensitivity to the effects of the control parameters exhibited by the proposed algorithm (*M*-BPSO) is reasonable. In fact, it can be clearly stated that the inertia weight parameter (ω) was not responsible for the exploration and exploitation activities of *M*-PSO, but the dynamically decreased of the particle's velocity limits.

5.3.2. Results for the high-scaled test problems

To further validate the computational effectiveness and efficiencies of *M*-BPSO and BPSO algorithms and study their reactions to the absence of some or all the control parameters in the particle's velocity update formula, they were tested using 6 scaled test problems with higher dimension. Presented in Tables (13) – (19) are the results obtained by all the variants of both algorithms optimizing these problems. In all the results, except Table 16 where $BPSO_4$ obtained better Best fitness and Mean fitness values, all the variants of *M*-PSO clearly outperformed their respective counterpart in all the test problems. In *Rastrigin* (Tables 14 and 15), *Ackley* and *Griewank* problems in all the tables, *M*-BPSO variants were able to obtain 100% success rate and global minimum, satisfying the success criteria which BPSO variants could not achieve. The results further revealed that BPSO can only obtain better results when all the control parameters are used in the particles' velocity updating formula, showing that it is sensitive to the parameters and that the inertia weight parameter (ω) was responsible for its exploration and exploitation activities (since the variants of BPSO that seems to perform better than their fellow variants, for example $BPSO_1$ and $BPSO_4$, contain the inertia weight parameter); whereas the inertia weight parameter has little or no effects on the variants of *M*-BPSO (since there are some of its variants without the inertial weight parameters, for example *M*-$BPSO_6$ and *M*-$BPSO_7$, which performed better than their fellow variants that have). Also, from the results, there are evidences of consistent successful performances by the variants of *M*-BPSO compared with those of BPSO.

From the results, it can be observed that it is only *M*-PSO_2 and *M*-$BPSO_3$ in Tables (14) and (15) that could obtain 100% success rate optimizing *Rastrigin* problem. On investigation it was experimentally discovered that all the *M*-BPSO variants could obtain 100% success rate for *Rastrigin* when the parameter δ was set to 0.15. This value was discovered, through further experiments, to be efficient for these variants optimizing the other problems with 100% success rate as well. With this discovery, it was clear that further experiments will prove the optimal value for δ to improve on the current performance of *M*-BPSO.

5.3.3. Comparison of *M*-BPSO with some existing optimization algorithms

When it became clear that *M*-BPSO could perform efficiently without the inertia weight parameter (δ), its variants without this parameter (*M*-$BPSO_2$, *M*-$BPSO_5$, *M*-$BPSO_6$ and *M*-$BPSO_7$) were selected for further comparisons with Modified Attractive-Repulsive PSO (MARPSO) [11] and Bioluminescent Swarm Optimization (BSO) algorithm [21]. The results for the two competing algorithms were obtained from the respective referenced literature. Presented in Table 20 are the results for MARPSO and the selected 4 *M*-BPSO variants while the results in Tables (21) – (24) are for BSO and the 4 *M*-BPSO variants. In these comparisons, δ was set to 0.15 as a result of the discovery explained before.

Table 20 shows the mean best fitness values obtained by the algorithms for all the 4 testing problems. The solution error tolerance, according to [11], was set to 1.0e-10. Meaning that, any solution obtained that was lower than the error tolerance was taken to be zero (global minimum). From the results, all the algorithms performed equally in *Ackley* (when the problem dimension was 20), *Griewank* (when the problem dimension was 100) and *Rastrigin* (when the problem dimensions were 50 and 100). But all the *M*-BPSO variants outperformed MARPSO in *Ackley* (when the problem dimensions were 50 and 100) and in *Griewank* (when the problem dimensions were 20 and 50); in *Rastrigin*, when the problem dimension was 20,

only 2 of *M*-BPSO variants could outperform MARPSO. This is a clear indication that *M*-BPSO is better-off in global search and local refinement operations than MARPSO. However, *M*-BPSO variants were outperformed by MARPSO in *Rosenbrock* across the three different problem dimensions. These results equally show that *M*-BPSO is more efficient optimizing multimodal problems.

In comparison with BSO, all the selected variants of *M*-BPSO performed better being able to obtain global minimum and better stability in *Griewank* problem across the problem dimensions as shown in Table 21. The same thing was repeated in Table 22 for *Rastrigin* problem except *M*-BPSO$_7$ that could not meet up when the problem dimension was 50. In Table 23, BSO was better solution accuracy, while *M*-BPSO was better in algorithm stability (better standard deviation). For the *Generalized Schaffer's f6* problem, when the problem dimension was 10, all the variants of *M*-BPSO obtained global minimum and better stability compared with BSO. When the dimension was 30, all the variants of *M*-BPSO except *M*-BPSO$_7$, performed better than BSO in solution quality only. However, none of the variants could succeed over BSO when the dimension was increased to 50. From all these results, it is still very clear that, though simpler in nature, *M*-BPSO algorithm has the capability of efficient performance without some or all the control parameters in the velocity update formula.

Table 6: Results of *B*-PSO$_1$ and *M*-BPSO$_1$ for the 4 non-scaled problems (*Parameters used in velocity formula* = ω, c_1, c_2, r_1, and r_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$
Best Fitness	**-9.999989e-01**	-9.999880e-01	0.000000e+00	0.000000e+00	9.983321e-02	**0.000000e+00**	**-1.867309e+02**	-1.867233e+02
Mean Fitness	**-9.999989e-01**	-9.998001e-01	3.497728e-03	**0.000000e+00**	9.983321e-02	**0.000000e+00**	-1.845845e+02	**-1.855964e+02**
Std. Dev.	**1.221245e-15**	2.408844e-04	4.663637e-03	**0.000000e+00**	7.076311e-18	**0.000000e+00**	1.502480e+01	**1.809705e+00**
Av. Iteration	*	*	712.72	**173.20**	*	**919.10**	69.67	*
SR (%)	0	0	64	**100**	0	**100**	**98**	0

Table 7: Results of *B*-PSO$_2$ and *M*-BPSO$_2$ for the 4 non-scaled problems (*Parameters used in velocity formula* = c_1, c_2, r_1, and r_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$
Best Fitness	-9.999788e-01	**-9.999989e-01**	2.325063e-05	**0.000000e+00**	9.983322e-02	**0.000000e+00**	**-1.867308e+02**	-1.867262e+02
Mean Fitness	**-9.995661e-01**	-9.994382e-01	2.928990e-04	**0.000000e+00**	1.030151e-01	**0.000000e+00**	**-1.867242e+01**	-1.847963e+02
Std. Dev.	**4.707143e-04**	7.567356e-04	2.307922e-04	**0.000000e+00**	6.221797e-03	**0.000000e+00**	**7.596182e-03**	2.432173e+00
Av. Iteration	*	*	*	182.84	*	939.44	2429	*
SR (%)	0	0	0	**100**	0	**100**	**2**	0

Table 8: Results of *B*-PSO$_3$ and *M*-BPSO$_3$ for the 4 non-scaled problems (*Parameters used in velocity formula* = ω, c_1 and c_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$
Best Fitness	**-9.999989e-01**	-9.999841e-01	0.000000e+00	0.000000e+00	9.983321e-02	**0.000000e+00**	**-1.867309e+02**	-1.867240e+02
Mean Fitness	-9.798395e-01	**-9.996697e-01**	3.362930e-03	**0.000000e+00**	9.983321e-02	**0.000000e+00**	-1.836850e+02	**-1.849065e+02**
Std. Dev.	1.399807e-01	**4.013006e-04**	4.556663e-03	**0.000000e+00**	1.952778e-17	**0.000000e+00**	1.277241e+01	**3.854876e+00**
Av. Iteration	*	*	263.89	174.68	*	832.02	389.32	*
SR (%)	0	0	18	**100**	0	**100**	88	0

Table 9: Results of *B*-PSO$_4$ and *M*-BPSO$_4$ for the 4 non-scaled problems (*Parameters used in velocity formula* = ω, r_1, and r_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$
Best Fitness	**-9.999989e-01**	-9.999978e-01	0.000000e+00	0.000000e+00	9.983321e-02	**0.000000e+00**	**-1.867309e+02**	-1.867052e+02
Mean Fitness	**-9.999989e-01**	-9.998524e-01	7.739937e-03	**0.000000e+00**	1.917962e-01	**0.000000e+00**	**-1.867309e+02**	-1.847786e+02
Std. Dev.	**1.221245e-15**	2.101345e-04	5.981072e-03	**0.000000e+00**	1.338715e-01	**0.000000e+00**	**7.605131e-14**	4.099844e+00
Av. Iteration	*	*	148.23	167.82	*	946.68	77.28	*
SR (%)	0	0	26	**100**	0	**100**	**100**	0

Table 10: Results of *B*-PSO$_5$ and *M*-BPSO$_5$ for the 4 non-scaled problems (*Parameters used in velocity formula* = r_1, and r_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$
Best Fitness	**-9.999936e-01**	-9.999887e-01	1.200820e-06	**0.000000e+00**	9.983594e-02	**0.000000e+00**	**-1.867307e+02**	-1.867269e+02
Mean Fitness	-9.991236e-01	**-9.994715e-01**	3.977075e-04	**0.000000e+00**	1.106470e-01	**5.493292e-134**	**-1.867190e+02**	-1.855022e+02
Std. Dev.	9.938748e-04	**4.613385e-04**	3.636144e-04	**0.000000e+00**	1.686063e-02	**3.719269e-133**	**1.282698e-02**	1.493956e+00
Av. Iteration	*	*	2639.00	**203.72**	*	**978.42**	*	*
SR (%)	0	0	2	**100**	0	**100**	0	0

Table 11: Results of *B*-PSO$_6$ and *M*-BPSO$_6$ for the 4 non-scaled problems (*Parameters used in velocity formula* = c_1 and c_2)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$
Best Fitness	**-9.999987e-01**	-9.999945e-01	4.941064e-08	**0.000000e+00**	9.983321e-02	**0.000000e+00**	**-1.867309e+02**	-1.867231e+02
Mean Fitness	-9.571209e-01	**-9.994552e-01**	3.833779e-06	**0.000000e+00**	9.983498e-02	**0.000000e+00**	-1.848939e+02	**-1.849871e+02**
Std. Dev.	1.588489e-01	**5.335085e-04**	6.687517e-06	**0.000000e+00**	1.051058e-05	**0.000000e+00**	9.329951e+00	**3.460685e+00**
Av. Iteration	*	*	1359.16	**173.20**	*	**913.24**	1540.59	*
SR (%)	0	0	90	**100**	0	**100**	**34**	0

Table 12: Results of *B*-PSO$_7$ and *M*-BPSO$_7$ for the 4 non-scaled problems (*Parameters used in velocity formula* = *none*)

Problem	Easom		Schaffer's f6		Salomon		Shubert	
Algorithm	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$
Best Fitness	-9.005780e-01	**-9.999899e-01**	9.715912e-03	**0.000000e+00**	2.080237e-01	**0.000000e+00**	-1.860328e+02	**-1.867299e+02**
Mean Fitness	-2.606330e-01	**-9.995850e-01**	1.944057e-02	**0.000000e+00**	5.581529e-01	**0.000000e+00**	-1.670202e+02	**-1.855301e+02**
Std. Dev.	3.234060e-01	**3.777628e-04**	1.421977e-02	**0.000000e+00**	2.198294e-01	**0.000000e+00**	1.813388e+01	**1.827537e+00**
Av. Iteration	*	*	*	**176.40**	*	**1005.18**	*	*
SR (%)	0	0	0	**100**	0	**100**	0	0

6. Conclusion

In this paper, the basic particle swarm optimization (BPSO) was modified with no additional complex computational efforts to form another PSO variant (*M*-BPSO). The modifications were inspired by the drawbacks of BPSO with respect to premature convergence, weak local search ability and the desire to make the algorithm simpler but more efficient. Instead of using the inertia weight parameter, *M*-BPSO uses a dynamically decreased particle velocity limits to balance its global and local search activities. This is an indication that the inertia

weight parameter may not always be necessary for PSO algorithms to work effectively. Also, it was discovered form the experiments that with proper modifications to some other parts of PSO algorithms, the acceleration coefficients and random factors may not to be necessary in the particle velocity updating equation to obtain global optimal solutions to optimization problems. With the extensive numerical simulations carried out to test the computational effectiveness and efficiencies of the different variants of the proposed algorithm, it was discovered that the strength of the algorithm lies on the ability to quickly explore the search space to locate a near optimal solution and then begin to exploit the neighbourhood for refinement of the result with the help of dynamically decreased velocity limits.

The proposed algorithm could not obtain global minimum for Rosenbrock problem, therefore further study is needed to find out the cause. More work is needed to obtain an appropriate or optimized value for the parameter δ because it is very important to the proposed algorithm. Another area worth investigating is the effects of using the velocity updating formula without each of the three components (inertia, cognitive and social) in turn, would have on the proposed algorithm. Finally, application of the algorithm to real-world problems needs to be investigated.

Table 13: Results of *B*-PSO$_1$ and *M*-BPSO$_1$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = ω, c_1, c_2, r_1, and r_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$	B-PSO$_1$	*M*-BPSO$_1$
Best Fitness	1.465494e-14	**4.440892e-16**	0.000000e+00	0.000000e+00	**2.395702e-29**	7.601640e-03	3.225468e-03	**1.436999e-06**	1.789493e+01	**0.000000e+00**	**1.371127e+00**	2.866977e+01
Mean Fitness	1.261062e-01	**1.936229e-15**	1.302722e-02	**0.000000e+00**	1.907781e+00	**8.762036e-02**	7.897280e-03	**2.326464e-05**	3.576993e+01	**8.255920e+00**	2.926283e+01	**2.869272e+01**
Std. Dev.	3.808949e-01	**1.753472e-15**	1.424598e-02	**0.000000e+00**	1.603408e+00	**8.436275e-02**	2.687338e-03	**1.776083e-05**	9.996110e+00	**1.340263e+01**	2.125632e+01	**9.794859e-03**
Av. Iteration	1129.04	143.56	763.75	155.78	640.5	*	*	2064.0	*	1456.79	*	*
SR (%)	90	**100**	40	100	4	0	0	**28**	0	**66**	0	0

Table 14: Results of *B*-PSO$_2$ and *M*-BPSO$_2$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = c_1, c_2, r_1, and r_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$	B-PSO$_2$	*M*-BPSO$_2$
Best Fitness	3.318493e+00	**4.440892e-16**	1.690147e+00	**0.000000e+00**	4.086332e-01	**1.146486e-02**	2.633765e-03	**1.348283e-06**	5.621073e+01	**0.000000e+00**	8.970409e+02	**2.868996e+01**
Mean Fitness	3.745884e+00	**3.215206e-15**	1.830453e+00	**0.000000e+00**	3.436966e+00	**2.612111e-02**	5.335727e-03	**2.164105e-05**	7.379167e+01	**0.000000e+00**	1.231009e+03	**2.869612e+01**
Std. Dev.	1.034941e-01	**1.471699e-15**	6.485999e-02	**0.000000e+00**	1.979661e+00	**1.701562e-02**	1.606380e-03	**1.810866e-05**	9.393737e+00	**0.000000e+00**	1.771140e+02	**1.545186e-03**
Av. Iteration	*	206	*	209.74	*	*	*	2172.0	*	909.4	*	*
SR (%)	0	**100**	0	**100**	0	0	0	**32**	0	**100**	0	0

Table 15: Results of *B*-PSO$_3$ and *M*-BPSO$_3$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = ω, c_1 and c_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$	B-PSO$_3$	*M*-BPSO$_3$
Best Fitness	1.941143e+00	**4.440892e-16**	1.053085e+00	**0.000000e+00**	2.007583e-01	**1.311926e-02**	2.543067e-02	**7.404291e-07**	2.590078e+01	**0.000000e+00**	1.915256e+02	**2.865088e+01**
Mean Fitness	2.793210e+00	**4.440892e-16**	1.148693e+00	**0.000000e+00**	2.508899e+00	**2.031907e-01**	7.404169e-02	**2.639914e-05**	4.469248e+01	**1.421085e-16**	4.345740e+02	**2.869382e+01**
Std. Dev.	5.910545e-01	**0.000000e+00**	6.096850e+00	**0.000000e+00**	2.005186e+00	**1.530039e-01**	3.418975e-02	**2.429079e-05**	1.401771e+01	**9.947598e-16**	2.657047e+02	**1.058605e-02**
Av. Iteration	*	104.06	*	110.3	*	*	*	2055.07	*	841	*	*
SR (%)	0	**100**	0	**100**	0	0	0	**30**	0	**100**	0	0

Table 16: Results of *B*-PSO$_4$ and *M*-BPSO$_4$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = ω, r_1, and r_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$	B-PSO$_4$	*M*-BPSO$_4$
Best Fitness	2.407427e+00	**4.440892e-16**	1.706413e-13	**0.000000e+00**	4.476413e-01	**4.838234e-03**	5.177056e-02	**7.665106e-07**	1.590765e+01	**0.000000e+00**	**4.052000e+00**	2.866791e+01
Mean Fitness	4.726894e+00	**2.362555e-15**	8.358146e-02	**0.000000e+00**	2.595458e+00	**1.478970e-01**	1.583305e-01	**2.873640e-05**	3.016323e+01	**9.325919e+00**	**2.433012e+01**	2.869451e+01
Std. Dev.	1.210246e+00	**1.770663e-15**	8.793192e-02	**0.000000e+00**	1.737213e+00	**1.077346e-01**	6.860169e-02	**2.517682e-05**	**9.267444e+00**	1.182169e+01	2.210057e+01	**1.085634e-02**
Av. Iteration	*	159.92	777.67	169.68	*	*	*	2069.17	*	1737.29	*	*
SR (%)	0	**100**	12	**100**	0	0	0	**24**	0	**56**	0	0

Table 17: Results of *B*-PSO$_5$ and *M*-BPSO$_5$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = r_1, and r_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$	B-PSO$_5$	*M*-BPSO$_5$
Best Fitness	3.344808e+00	**4.440892e-16**	1.481730e+00	**0.000000e+00**	4.798298e-01	**1.394459e-02**	3.298895e-03	**7.663106e-07**	5.995154e+01	**0.000000e+00**	8.185073e+02	**2.869521e+01**
Mean Fitness	3.964400e+00	**3.854694e-15**	2.034272e+00	**0.000000e+00**	3.244917e+00	**2.887764e-02**	6.407808e-03	**2.823978e-05**	8.641181e+01	**1.286435e-05**	1.901100e+03	**2.869590e+01**
Std. Dev.	1.762065e-01	**6.961869e-16**	1.640075e-01	**0.000000e+00**	1.881748e+00	**2.789996e-03**	1.751897e-03	**2.197213e-05**	1.297889e+01	**9.005045e-05**	6.210673e+02	**5.868816e-04**
Av. Iteration	*	277.04	*	274.68	*	*	*	2256.0	*	1229.39	*	*
SR (%)	0	**100**	0	**100**	0	0	0	**26**	0	**98**	0	0

Table 18: Results of *B*-PSO$_6$ and *M*-BPSO$_6$ for the 6 scaled benchmark problems (*Parameters used in velocity formula* = c_1 and c_2)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$	B-PSO$_6$	*M*-BPSO$_6$
Best Fitness	1.774445e+00	**4.440892e-16**	1.091267e+00	**0.000000e+00**	3.969549e-02	**1.485857e-02**	2.727736e-03	**1.826915e-06**	3.425003e+01	**0.000000e+00**	1.833782e+02	**2.866868e+01**
Mean Fitness	2.771661e+00	**9.414691e-16**	1.312471e+00	**0.000000e+00**	2.292549e+00	**1.040067e-01**	5.931060e-03	**2.735918e-05**	5.564719e+01	**9.576085e-01**	7.435443e+02	**2.869606e+01**
Std. Dev.	4.115397e-01	**1.232746e-15**	1.182277e-01	**0.000000e+00**	1.605766e+00	**1.191236e-01**	2.276012e-03	**2.420559e-05**	1.615439e+01	**6.703259e+00**	4.269498e+02	**7.175068e-03**
Av. Iteration	*	115.32	*	114.38	*	*	*	2117.2	*	905.08	*	*
SR (%)	0	**100**	0	**100**	0	0	0	**30**	0	98	0	0

Table 19: Results of *B*-PSO$_7$ and *M*-BPSO$_7$ for the 6 scaled benchmark problems (*Parameters used in velocity formula = none*)

Problem	Ackley		Griewank		Levy		Noisy Quatic		Rastrigin		Rosenbrock	
Algorithm	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$	B-PSO$_7$	*M*-BPSO$_7$
Best Fitness	6.408485e+00	**4.440892e-16**	6.394065e+00	**0.000000e+00**	2.733269e+00	**1.390444e-02**	2.987741e-02	**1.452023e-06**	1.418611e+02	**0.000000e+00**	3.412631e+04	**2.867724e+01**
Mean Fitness	9.025067e+00	**3.641532e-15**	1.185077e+01	**0.000000e+00**	7.539175e+00	**6.907410e-02**	8.506966e-02	**3.242492e-05**	1.862238e+02	**1.995969e+01**	1.232955e+05	**2.869711e+01**
Std. Dev.	1.159080e+00	**1.065814e-15**	3.358921e+00	**0.000000e+00**	3.967178e+00	**9.854786e-02**	5.146707e-02	**3.143463e-05**	2.269556e+01	**3.175845e+01**	9.131989e+04	**6.736141e-03**
Av. Iteration	*	134.22	*	139.20	*	*	*	2367.7	*	1433.06	*	*
SR (%)	0	**100**	0	**100**	0	0	0	**20**	0	**66**	0	0

Table 20: Comparison between MARPSO and proposed variants

Problem Dimension	Algorithm	Problem			
		Ackley	Griewank	Rastrigin	Rosenbrock
20	MARPSO	0.00e+00	4.03e-03	0.00e+00	**0.13**
	M-BPSO$_2$	0.00e+00	**0.00e+00**	0.00e+00	1.879e+01
	M-BPSO$_5$	0.00e+00	**0.00e+00**	0.00e+00	1.879e+01
	M-BPSO$_6$	0.00e+00	**0.00e+00**	1.33e-03	1.879e+01
	M-BPSO$_7$	0.00e+00	**0.00e+00**	3.11e-07	1.879e+01
50	MARPSO	2.39e-10	1.97e-04	0.00e+00	**1.28**
	M-BPSO$_2$	**0.00e+00**	**0.00e+00**	0.00e+00	4.850e+01
	M-BPSO$_5$	**0.00e+00**	**0.00e+00**	0.00e+00	4.850e+01
	M-BPSO$_6$	**0.00e+00**	**0.00e+00**	0.00e+00	4.851e+01
	M-BPSO$_7$	**0.00e+00**	**0.00e+00**	0.00e+00	4.850e+01
100	MARPSO	3.99e-09	0.00e+00	0.00e+00	**16.93**
	M-BPSO$_2$	**0.00e+00**	0.00e+00	0.00e+00	9.802e+01
	M-BPSO$_5$	**0.00e+00**	0.00e+00	0.00e+00	9.800e+01
	M-BPSO$_6$	**0.00e+00**	0.00e+00	0.00e+00	9.809e+01
	M-BPSO$_7$	**0.00e+00**	0.00e+00	0.00e+00	9.803e+01

Table 21: Comparison between BSO and the proposed variants in *Griewank* problem

Dimension	Measurement	BSO	*M*-BPSO$_2$	*M*-BPSO$_5$	*M*-BPSO$_6$	*M*-BPSO$_7$
10	**Mean Fitness**	0.03465	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.02183	**0.00000**	**0.00000**	**0.00000**	**0.00000**
30	**Mean Fitness**	0.02628	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.02542	**0.00000**	**0.00000**	**0.00000**	**0.00000**
50	**Mean Fitness**	0.02919	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.01673	**0.00000**	**0.00000**	**0.00000**	**0.00000**

Table 22: Comparison between BSO and the proposed variants in *Rastrigin* problem

Dimension	Measurement	BSO	*M*-BPSO$_2$	*M*-BPSO$_5$	*M*-BPSO$_6$	*M*-BPSO$_7$
10	**Mean Fitness**	0.00005	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.00004	0.00000	0.00000	0.00000	0.00000
30	**Mean Fitness**	0.14368	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.38712	0.00000	0.00000	0.00000	0.00000
50	**Mean Fitness**	0.32219	**0.00000**	**0.00000**	**0.00000**	1.80655
	Std. Dev.	0.80927	0.00000	0.00000	0.00000	17.9749

Table 23: Comparison between BSO and the proposed variants in *Rosenbrock* problem

Dimension	Algorithm	BSO	*M*-BPSO$_2$	*M*-BPSO$_5$	*M*-BPSO$_6$	*M*-BPSO$_7$
10	**Mean Fitness**	**0.72827**	8.80651	8.80144	8.73759	8.78180
	Std. Dev.	1.52126	0.09826	0.09900	0.20571	0.12739
30	**Mean Fitness**	**27.0083**	28.69899	28.6980	28.70292	28.70190
	Std. Dev.	1.75217	0.00657	0.00512	0.01177	0.00958
50	**Mean Fitness**	**47.0415**	48.49819	48.4961	48.51395	48.50054
	Std. Dev.	0.79140	0.00340	0.00145	0.02086	0.00620

Table 24: Comparison between BSO and the proposed variants in *Generalized Schaffer's f6* problem

Dimension	Algorithm	BSO	*M*-BPSO$_2$	*M*-BPSO$_5$	*M*-BPSO$_6$	*M*-BPSO$_7$
10	**Mean Fitness**	0.07870	**0.00000**	**0.00000**	**0.00000**	**0.00000**
	Std. Dev.	0.02445	0.00000	0.00000	0.00000	0.00000
30	**Mean Fitness**	0.50525	**0.08966**	**0.22178**	**0.14591**	0.78340
	Std. Dev.	0.18516	0.50456	0.87080	0.72556	1.61286
50	**Mean Fitness**	**1.39567**	7.85840	9.92779	7.62957	10.92979
	Std. Dev.	0.55424	5.09182	3.52718	4.94611	1.79961

References

[1] Arasomwan, A. M. and Adewumi, A. O. On the Performance of Linear Decreasing Inertia Weight Particle Swarm Optimization for Global Optimization, *The Scientific World Journal*, 2013: 1-12

[2] Chen, F., Sun, X. and Wei, D. Inertia weight particle swarm optimization with Boltzmann exploration, *Seventh International Conference on Computational Intelligence and Security (CIS),* 2011. pp. 90-95.

[3] Chauhan, P., Deep, K. and Pant, M. Novel inertia weight strategies for particle swarm optimization, *Memet. Comput,* 2013; 5: 229-251.

[4] Chen, G. Huang, X. Jia, J. and Min., Z. Natural exponential Inertia Weight strategy in particle swarm optimization, *Sixth World Congress on Intelligent Control and Automation, WCICA*, June 21 - 23, Dalian, China, 2006; 1: 3672–3675.

[5] Clerc, M. and Kennedy, J. The particle swarm - explosion, stability, and convergence in multidimensional complex space, *IEEE Trans on Evol Comp,* 2002; 6: 58-73.

[6] Wen, L., Xi, Z. The research of PSO algorithms with non-linear time-decreasing inertia weight, *7th World Congress on Intelligent Control and Automation,* 2008; pp. 4002-4005.

[7] Eberhart R. C. and Kennedy J. A new optimizer using particle swarm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95,* 1995, pp. 39-43.

[8] Eberhart, R. C. and Shi, Y. Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 Congress* on *Evolutionary Computation,* Korea, 2002; 1: 94–100.

[9] Evers, G. I. *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization*. MSc. Thesis. University of Texas-Pan American, 2009.

[10] Feng, Y., Teng, G.F., Wang, A.X. and Yao, Y.M. Chaotic inertia weight in particle swarm optimization, *IEEE Second International Conference on Innovative Computing, Information and Control*, 2007, pp. 475.

[11] Guochao, N., Baodi, C. and Jianchao, Z. Repulsive Particle Swarm Optimization based on new diversity, in *Control and Decision Conference (CCDC),* 2010, pp. 815-819.

[12] Jiang, M., Luo, Y. P. and Yang, S. Y. Particle swarm optimization - stochastic trajectory analysis and parameter selection, In , Felix T.S. Chan and Manoj Kumar Tiwari (Ed.), *Swarm Intelligence*: *Focus on Ant and Particle Swarm Optimization*. InTech Publisher, 2007, pp. 179-198.

[13] Jihong, S. and Wensuo, Y. Improvement of original particle swarm optimization algorithm based on simulated annealing algorithm, *Eighth International Conference on Natural Computation (ICNC),* 2012, pp. 777-781.

[14] Karaboga, D., and Akay B. A Comparative Study of Artificial Bee Colony Algorithm. *Appl. Math. Comput.*, 2009; 214: 108-132.

[15] Kennedy, J. and Eberhart, R.C. Particle Swarm Optimization, *IEEE international conference on neural networks*, 4 Perth, Australia, 1995, pp. 1942–1948.

[16] Kentzoglanakis, K. and Poole, M. Particle swarm optimization with an oscillating inertia weight, *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009, pp. 1749–1750.

[17] Li, H.R. and Gao, Y.L. Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation, *Second International Conference on Information and Computing Science*, 2009, pp. 66–69.

[18] Mahamed G.H. Omran, Using Opposition-based Learning with Particle Swarm Optimization and Barebones Differential Evolution, In: Aleksandar Lazinica (Ed.), *Particle Swarm Optimization*, InTech Publisher, 2009, pp. 373-384.

[19] Malik, R.F., Rahman, T.A., Hashim, S.Z.M. and Ngah, R. New particle swarm optimizer with sigmoid increasing inertia weight, *Internat. J. of Comp. Sc .& Secur.*, 2007; 1(2): 35-44.

[20] Nickabadi, A., Ebadzadeh, M. M. and Safabakhsh, R. A novel particle swarm optimization algorithm with adaptive inertia weight, *Appl. soft comp.*, 2011; 11: 3658-3670.

[21] Oliveira, D. R., Parpinelli, R. S. and Lopes, H. S. Bioluminescent Swarm Optimization Algorithm, In: Eisuke Kita (Ed.) *Evolutionary Algorithms*, InTech Publisher, 2011, pp. 69-84.

[22] Shen, X., Chi, Z., Yang, J., Chen, C., and Chi, Z. Particle swarm optimization with dynamic adaptive inertia weight. *International Conference on Challenges in Environmental Science and Computer Engineering, IEEE*, 2010, pp. 287-290.

[23] Shi, Y. H., Eberhart, R.C. A modified particle swarm optimizer. *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, May 4-9, 1998, pp. 69-73.

[24] Shi, Y. and Eberhart, R. Parameter selection in particle swarm optimization, In: V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, (Eds). *Evolutionary Programming VII*. Springer Berlin Heidelberg, 1998; 1447: 591-600.

[25] Shi, Y.H., and Eberhart, R.C. Fuzzy adaptive particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation*, Korea, 2001; 1: 101-106.

[26] Sun, J., Lai, C-H, and Wu, X-J. Particle swarm optimization: classical and quantum perspectives, CRC press, New York, 2011.

[27] Trelea, I.C. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information. Processing Letter*, 2003; 85: 317-325.

[28] Yuhui, S. and Eberhart, R.C. Empirical study of particle swarm optimization, *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99.*, 1999; 3, pp. 1945-1950.

[29] Liu, B., Wang, L., Jin, Y., Tang, F. and Huang, D. Improved particle swarm optimization combined with chaos, *Chaos, Solution and Fractals*, 2005; 25: 1261-1271.

[30] Su, T-J. Cheng, J-C. and Sun, Y-D. Particle swarm optimization with time-varying acceleration coefficients based on cellular neural network for color image noise cancellation, *Sixth international conference on digital telecommunications*. ICDT 2011, pp. 109-115.

[31] Selvakumar, A. I. and Thanushkodi, K. A new particle swarm optimization solution to nonconvex economic dispatch Problems. *IEEE Transactions on Power Systems*, 2007; 22(1): 42-51.

[32] Selvakumar, A. I. and Thanushkodi, K. Anti-predatory particle swarm optimization: Solution to nonconvex economic dispatch problems, *Electric Power Systems Research*, 2008; 78: 2-10.

[33] Guo, W. Chen, G. and Feng, X. A new strategy of acceleration coefficients for particle swarm optimization, *Proceedings of the 10th IEEE International Conference on Computer Supported Cooperative Work in Design*, 2006, pp. 1-5.

[34] Chetty, S. and A. O. Adewumi A.O. Three new stochastic local search algorithms for continuous optimization problems," *Computational Optimization and Applications*, 2013 53(3): 675-721.

[35] Sawyerr, B. A., Ali, M.M. and Adewumi, A.O. A comparative study of some real-coded genetic algorithms for unconstrained global optimization. *Optimization Methods and Software*, 2011; 26(6): 945–970.

Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 798129, 23 pages
http://dx.doi.org/10.1155/2014/798129

Research Article

Improved Particle Swarm Optimization with a Collective Local Unimodal Search for Continuous Optimization Problems

Martins Akugbe Arasomwan and Aderemi Oluyinka Adewumi

School of Mathematics, Statistics, and Computer Science, University of Kwazulu-Natal South Africa, Private Bag X54001, Durban 4000, South Africa

Correspondence should be addressed to Aderemi Oluyinka Adewumi; laremtj@gmail.com

Received 31 October 2013; Accepted 29 December 2013; Published 25 February 2014

Academic Editors: T. Chen, Q. Cheng, and J. Yang

A new local search technique is proposed and used to improve the performance of particle swarm optimization algorithms by addressing the problem of premature convergence. In the proposed local search technique, a potential particle position in the solution search space is collectively constructed by a number of randomly selected particles in the swarm. The number of times the selection is made varies with the dimension of the optimization problem and each selected particle donates the value in the location of its randomly selected dimension from its personal best. After constructing the potential particle position, some local search is done around its neighbourhood in comparison with the current swarm global best position. It is then used to replace the global best particle position if it is found to be better; otherwise no replacement is made. Using some well-studied benchmark problems with low and high dimensions, numerical simulations were used to validate the performance of the improved algorithms. Comparisons were made with four different PSO variants, two of the variants implement different local search technique while the other two do not. Results show that the improved algorithms could obtain better quality solution while demonstrating better convergence velocity and precision, stability, robustness, and global-local search ability than the competing variants.

1. Introduction

Optimization comes to focus when there are needs to plan, take decisions, operate and control systems, design models, or seek optimal solutions to varieties of problems faced from day to day by different people. A number of these problems, which can be formulated as continuous optimization problems, are often approached with limited resources. Dealing with such problems, most especially when they are large scale and complex, has attracted the development of different nature-inspired optimization algorithms. These algorithms display problem-solving capabilities for researchers to solve complex and challenging optimization problems with many success stories. Swarm-based techniques are a family of nature-inspired algorithms and are population-based in nature; they are also known as evolutionary computation techniques. Particle swarm optimization (PSO) technique is a member of swarm-based techniques which is capable of producing low cost, fast, and robust solutions to several complex optimization problems. It is a stochastic, self-adaptive, and problem-independent optimization technique and was originally proposed in 1995 by Eberhart and Kennedy as simulation of a flock of bird or the sociological behavior of a group of people [1, 2]. From the time this concept was brought into optimization, it has been used extensively in many fields which include function optimization and many difficult real-world optimization problems [3–5].

PSO technique was initially implemented with few lines of codes using basic mathematical operations with no major adjustment needed to adapt it to new problems and it was almost independent of the initialization of the swarm [6]. It needs few parameters to operate with for successful and efficient behavior in order to obtain quality solutions. To implement this technique, a number of particles, which are characterized by positions and velocities, called swarm are required to be randomly distributed in a solution search space depending on the boundaries defined for the design variables of the problem being optimized. The number of

design variables determines the dimensionality of the search space. If d-dimensional space is considered, the position and velocity of each particle are represented as the vectors $X_i = (x_{i1}, \ldots, x_{id})$ and $V_i = (v_{i1}, \ldots, v_{id})$, respectively. Every particle has a memory of its personal experience which is communicated to all reachable neighbours in the search space to guide the direction of movement of the swarm. Also, the quality of each particle (solution) is determined by the objective function of the problem being optimized and the particle with best quality is taken as the global solution towards which other particles will converge. The common practice is for the technique to maintain a single swarm of particles throughout its operation. This process of seeking optimal solution involves the adjustments of the position and velocity of each particle in each iteration using

$$V_i(t+1) = \omega V_i(t) + \text{coeff}_1\left(P_i - X_i\right) + \text{coeff}_2\left(P_g - X_i\right), \tag{1}$$

$$X_i(t+1) = X(t) + V_i(t+1). \tag{2}$$

In (1), P_i and P_g are vectors representing the ith particle personal best and swarm global best positions, respectively; $\text{coeff}_1 = c_1 r_1$ and $\text{coeff}_2 = c_2 r_2$; c_1 and c_2 are acceleration factors known as cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of P_i and P_g; r_1 and r_2 are random numbers between 0 and 1; t is iteration index. The symbol ω is the inertia weight parameter which was introduced into the original PSO in [7]. The purpose of its introduction was to help the PSO algorithm balance its global and local search activities.

There are possibilities of the positions and velocities of the particles in the swarm increasing in value beyond necessary when they are updated. As a measure, the positions are clamped in each dimension to the search range $[X_{\min}, X_{\max}]$ of the design variables, where $X_{\min}$ and $X_{\max}$ represent the lower and upper bounds of a particle's position, respectively, while their velocities are controlled to be within a specified range $[V_{\min}, V_{\max}]$, where $V_{\min}$ and $V_{\max}$ represent the lower and upper bounds of a particle's velocity, respectively. The idea of velocity clamping which was introduced by [1, 2, 8] and extensively experimented with in [9] has led to significant improvement as regards the performance of PSO. This is so because the particles could concentrate, taking reasonably sized steps to search through the search space rather than bouncing about excessively. A major feature that characterizes an efficient optimization algorithm is the ability to strike a balance between local and global search. Global search involves the particles being able to advance from a solution to other parts of the search space and locate other promising candidates while local search means that the particle is capable of exploiting the neighbourhood of the present solution for other promising candidates. In PSO, as the rate of information sharing increases among the particles they migrate towards the same direction and region in the search space. If any of the particles could not locate any better global solution after some time, they will eventually converge about the existing one which may not be the global minimum due to lack of exploration power; this is known as premature convergence. This type of behaviour is more likely when the swarm of particles is overconcentrated. It could also occur when the optimization problem is of high dimension and/or nonconvex. One of the possible ways to prevent this premature convergence is to embed a local search technique into PSO algorithm to help improve the quality of each solution by searching its neighbourhood. After the improvement, better information is communicated among the particles thereby increasing the algorithm's ability to locate better global solution in course of optimization. Hill climbing, modified Hooke and Jeeves, gradient descent, golden ratio, Stochastic local search, adaptive local search, local interpolation, simulated annealing, and chaotic local search are different local search techniques that have been combined with PSO to improve its local search ability [10–18].

In this paper, a different local search technique was proposed to harness the global search ability of PSO and improve on its local search efforts. This technique is based on the collective efforts of randomly selected (with replacement) particles a number of times equal to the size of the problem dimension. When a particle is selected, it is made to contribute the value in the position of its randomly selected dimension from its personal best. The contributed values are then used to form a potential global best solution which is further refined. This concept could offer PSO the ability to enhance its performance in terms of convergence speed, local search ability, robustness, and increased solution accuracy. The local search technique was hybridized with two of the existing PSO variants, namely, random inertia weight PSO (RIW-PSO) and linear decreasing inertia weight PSO (LDIW-PSO), to form two new variants. Numerical simulations were performed to validate the efficiencies of each of them and some statistical analyses were performed to ascertain any statistically significant difference in performance between the proposed variants and the old ones. From the results obtained it was shown that the proposed variants are very efficient.

In the sections that follow, RIW-PSO and LDIW-PSO are briefly described in Section 2; the motivation and description of the proposed local search technique are presented in Section 3 while the improved PSO with local search technique is described in Section 4. Numerical simulations are performed in Section 5 and Section 6 concludes the paper.

2. The Particle Swarm Optimization Variants Used

Two PSO variants were used to validate the proposed improvement of the performance of PSO technique. The variants are LDIW-PSO and RIW-PSO. These were chosen because of the evidence available in the literature that they are less efficient in optimizing many continuous optimization problems [19–21]. These variants are succinctly described below.

2.1. PSO Based on Linear Decreasing Inertia Weight (LDIW-PSO). This variant was proposed in [9] after the inertia

weight parameter was introduced into the original PSO by [7]. It implements the linear decreasing inertia weight strategy represented in (3) which decreases from some high which facilitates exploration to a low value which on the other hand promotes exploitation. This greatly improved the performance of PSO. LDIW-PSO does global search at the beginning and converges quickly towards optimal positions but lacks exploitation power [9] and the ability required to jump out of the local minimum most especially when being in the multimodal landscape. Some improvements on LDIW-PSO exist in the literature [6, 9, 22]:

$$\omega_{\mathrm{i}} = \left(\omega_{\mathrm{start}} - \omega_{\mathrm{stop}}\right)\left(\frac{\mathrm{MAX}_{\mathrm{itr}} - \mathrm{i}}{\mathrm{MAX}_{\mathrm{itr}}}\right) + \omega_{\mathrm{stop}}, \tag{3}$$

where ω_{start} and ω_{stop} are the initial and final values of inertia weight, i is the current iteration number, $\mathrm{MAX}_{\mathrm{itr}}$ is the maximum iteration number, and $\omega_{\mathrm{i}} \in [0, 1]$ is the inertia weight value in the ith iteration. Apart from the problem of premature convergence, this variant was found inefficient in tracking a nonlinear dynamic system because of the difficulty in predicting whether exploration (a larger inertia weight value) or exploitation (a smaller inertia weight) will be better at any given time in the search space of the nonlinear dynamic system [23].

2.2. PSO Based on Random Inertia Weight (RIW-PSO). Due to the improved performance of PSO when the constant inertia weight was introduced into it [7], a new era of research was indirectly initiated and this has attracted the attentions of many researchers in the field. The inefficiency of linear decreasing inertia weight, which linearly decreases from 0.9 to 0.4, in tracking a nonlinear dynamic system prompted the introduction of RIW which randomly varies within the same range of values. Random adjustment is one of the strategies that have been proposed to determine the inertia weight value to further improve on the performance of PSO. This strategy is nonfeedback in nature and the inertia weight takes different value randomly at each iteration, from a specified interval. In line with this, random inertia weight strategy represented in (4) was introduced into PSO by [23] to enable the algorithm track and optimize dynamic systems. In the equation, rand() is a uniform random number in the interval $[0, 1]$ which make the formula generate a number randomly varying between 0.5 and 1.0, with a mean value of 0.75. When c_1 and c_2 are set to 1.494, the algorithm seems to demonstrate better optimizing efficiency. The motivation behind the selection of these values was Clerc's constriction factor [23]:

$$\omega = 0.5 + \frac{\mathrm{rand}\,()}{2}. \tag{4}$$

Not much is recorded in the literature regarding the implementation of this variant of PSO. Some of the few implementations found in the literature are recorded in [6, 20–22].

3. Proposed Local Search Technique

The basic principle underlying the optimizing strategy of PSO technique is that each particle in the swarm communicates their discoveries to their neighbours and the particle with the best discovery attracts others. While this strategy looks very promising, there is the risk of the particles being susceptible to premature convergence, especially when the problem to be optimized is multimodal and high in dimensionality. The reason is that the more the particles share their discoveries among themselves, the higher their identical behaviour is until they converge to the same area in the solution search space. If none of the particle could discover better global best, after some time all the particles will converge about the existing global best which may not be the global minimizer.

One of the motivations for this local search technique is the challenge of premature convergence associated with PSO technique which affects its reliability and efficiency. Another motivation is the decision-making strategy used by the swarm in searching for optimal solution to optimization problems. The decision is dictated by a single particle in the swarm; that is, other particles follow the best particle among them to search for better solution. Involving more than one particle in the decision making could lead to a promising region in the search space where optimal solution could be obtained.

The description of the local search technique is as follows: after all the particles have obtained their various personal best positions, each particle has an equal chance of being selected to contribute its idea towards how a potential location in the search space where better global best could be obtained. As a result, a number of particles equal to the dimension of the problem being optimized are randomly selected (with replacement). Each selected particle contributes an idea by donating the value in the location of its randomly selected dimension from its personal best. All the ideas contributed by the selected particles are collectively used (hybridized) to construct a potential solution in the solution search space. After constructing the potential solution, some searches are locally done around its neighbourhood with the hope of locating a better solution in comparison with the current global solution. If a better solution is found, it is then used to replace the current global solution; otherwise no replacement is made.

In this local search, the potential new position is denoted by $\overrightarrow{y}$ and is sampled from the neighbourhood of the collectively constructed potential global solution represented as $\overrightarrow{P}$ by

$$\overrightarrow{y} \longleftarrow \overrightarrow{P} + \overrightarrow{a}, \tag{5}$$

where $\overrightarrow{a} \sim U[-\overrightarrow{r}, \overrightarrow{r}]$ is a random vector picked uniformly from the range $[-\overrightarrow{r}, \overrightarrow{r}]$ and $\overrightarrow{r}$ is the search radius which is initially set to max$R$ (maximum radius for local search). The local search technique moves from position $\overrightarrow{P}$ to position $\overrightarrow{y}$ when there is improvement to the fitness. If there is no improvement on the fitness of $\overrightarrow{P}$ by $\overrightarrow{y}$, the search radius is linearly decreased by multiplying it with a factor q using

$$\begin{aligned} \overrightarrow{r} &\longleftarrow q \times \overrightarrow{r}, \\ q &\longleftarrow (\max R - \min R) \times \frac{t}{\max T} + \min R, \end{aligned} \tag{6}$$

$\overrightarrow{r}, \overrightarrow{a}, \overrightarrow{y} \leftarrow$ new arrays, each of length Dim
$\overrightarrow{r} \leftarrow \max R$
$t \leftarrow 0$
While $(t < \max T)$ do
  $t \leftarrow t + 1$
  $\overrightarrow{a} \leftarrow U(-\overrightarrow{r}, \overrightarrow{r})$
  for $j \leftarrow 1$ to problem Dimension
    randomly select any particle i
    randomly select a dimension d from the personal best P of the selected particle i
      $\overrightarrow{y}^j \leftarrow \overrightarrow{P}_i^d + \overrightarrow{a}^j$
  end for
  validate for search space boundary
  If $f(\overrightarrow{y}) < g$Fit
    $\overrightarrow{g\text{Pos}} \leftarrow \overrightarrow{y}$
    gFit $\leftarrow f(\overrightarrow{y})$
  else
    $q \leftarrow (\max R - \min R) \times \dfrac{t}{\max T} + \min R$
    $\overrightarrow{r} \leftarrow q \times \overrightarrow{r}$
  end if
end while
Return $\overrightarrow{g\text{Pos}}$ and gFit

Algorithm 1: Collective local unimodal search.

Begin PSO_{CLUS} Algorithm
Step 1. **Definition Phase**
(1.1) function to optimize as f
(1.2) Parameter
(1.2.1) swarm size
(1.2.2) problem dimension
(1.2.3) solution search space
(1.2.4) particle velocity range
Step 2. **Initialized phase**
For all particles randomly initialized in search space
(2.1) position $x_i \leftarrow (x_{i1}, \ldots, x_{id})$
(2.2) velocity $v_i \leftarrow (v_{i1}, \ldots, v_{id})$,
(2.3) $p\text{best}_i \leftarrow (x_{i1}, \ldots, x_{id})$
(2.4) $g$best$\leftarrow$ best of $p\text{best}_i$
(2.5) evaluate $f(x_i)$ using objective function of problem
Step 3. **Operation Phase**
Repeat until a stopping criterion is satisfied
(3.1). Compute inertia weight using any inertia weight formula
(3.2). For each particle i
(3.2.1). update v_i for particle using (1)
(3.2.2). validate for velocity boundaries
(3.2.3). update x_i for particle using (2)
(3.2.4). validate for position boundaries
(3.2.5). If $f(x_i) < f(p\text{best}_i)$ then $p\text{best}_i \leftarrow x_i$
(3.3). $g$best $\leftarrow$ best of $p$best$i$
(3.4). Implement local search using CLUS in Algorithm 1
Step 4. **Solution Phase**
(4.1). $x^* \leftarrow g$best
(4.2). $f^* \leftarrow f(g\text{best})$
(4.3). Return x^* and f^*
End PSO_{CLUS} Algorithm

Algorithm 2: Algorithm for PSO_{CLUS}.

Table 1: Parameter settings for experiment.

Parameter	$\omega_{\min}$	$\omega_{\max}$	$c_1 = c_2$	$V_{\min}$	$V_{\max}$	min$R$	max$R$	max$T$
Value	0.9	0.4	1.494	$0.05 * X_{\min}$	$0.05 * X_{\max}$	0.01	2.0	100

where max$T$ is the maximum number of times the neighbourhood of $\overrightarrow{P}$ is to be sampled, t is the current time the neighbourhood is being sampled, and min$R$ is the minimum radius for the local search.

This proposed local search technique has been named collective local unimodal search (CLUS) technique. It has some trace of similarity in operation with local unimodal sampling (LUS) technique [24]. But they are quite different in the sense that, while LUS randomly picks a potential solution from the entire population, CLUS constructs a potential solution using the collective efforts of a randomly selected number of particles from the swarm. Also, CLUS uses a linear method to decrease the search radius (step size) in the neighbourhood of the potential solution which is different from the method applied by LUS during optimization. The CLUS technique is presented in Algorithm 1. In the technique, gFit and $\overrightarrow{g\text{Pos}}$ represent the current global fitness value and its corresponding position in the search space.

4. Improved PSO with Collective Unimodal Local Search (PSO_{CLUS})

The RIW-PSO increases convergence in early iterations and does more of global search activities but soon gets stuck in local optima because of lack of local search ability. Also, LDIW-PSO does global search at earlier part of its iteration but lacks enough momentum to do local search as it gets towards its terminal point of execution. The aim of this paper is to make a general improvement on the performance of PSO which can be applied to any of its variants. To achieve this, the two PSO variants were hybridized with the proposed collective local unimodal search (CLUS) technique which takes advantage of their global search abilities to do some neighbourhood search for better results. The improved PSO algorithm is presented in Algorithm 2.

5. Numerical Simulations

In this section, the improved algorithm (PSO_{CLUS}) was implemented using the inertia weight strategy of RIW-PSO and the variant was labeled R-PSO_{CLUS}. It was also implemented using the inertia weight strategy of LDIW-PSO and the variant was labeled L-PSO_{CLUS}. The performances of R-PSO_{CLUS} and L-PSO_{CLUS} were experimentally tested against those of RIW-PSO and LDIW-PSO, respectively. The maximum number of iterations allowed was 1000 for problems with dimensions less than or equal to 10, 2000 for 20-dimensional problems, and 3000 for 30-dimensional problems. A swarm size of 20 was used in all the experiments and twenty-five independent runs were conducted to collect data for analysis. The termination criteria for all the algorithms were set to be as maximum number of iterations relative to the problems' dimensions. A run, in which an algorithm is able to satisfy the set success criteria (see Table 1) before or at the maximum iteration, is considered to be successful. To further prove the efficiency of the proposed local search technique, the proposed PSO variants were also compared with some existing PSO variants hybridized with different local search techniques. They are PSO with golden ratio local search [15] and PSO with local interpolation search [18]. A total of 6 different experiments were conducted.

(i) R-PSO_{CLUS} was compared with PSO with golden ratio local search (GLSPSO);

(ii) R-PSO_{CLUS} was compared with PSO with local interpolation search (PSOlis);

(iii) R-PSO_{CLUS} was compared with RIW-PSO;

(iv) L-PSO_{CLUS} was compared with PSO with golden ratio local search (GLSPSO);

(v) L-PSO_{CLUS} was compared with PSO with local interpolation search (PSOlis);

(vi) L-PSO_{CLUS} was compared with LDIW-PSO.

The application software was developed in Microsoft Visual C# programming language.

5.1. Test Problems. A total of 21 problems were used in the experiments. These problems have different degrees of complexity and multimodality which represents diverse landscapes enough to cover many of the problems which can arise in global optimization problems. Shown in Table 2 are the problems dimensions, optimal fitness values, and success thresholds. Presented in Table 3 are the definitions, characteristics (US: unimodal separable, UN: unimodal nonseparable, MS: multimodal separable, and MN: multimodal nonseparable), and search ranges of the problems. More details on the benchmark problems can be found in [22, 25–27].

5.2. Parameter Setting. The additional parameters that were set in the experiment are inertia weight threshold for LDIW-PSO ($\omega_{\min}$ and $\omega_{\max}$), acceleration coefficients (c_1 and c_2), velocity thresholds ($V_{\min}$ and $V_{\max}$), minimum radius (min$R$), and maximum radius (max$R$) for local search as well as the maximum number of neighbourhood sampling (max$T$) during the local search. The respective settings of these parameters are shown in Table 1. The parameters r_1 and r_2 were randomly generated using the uniform random number generator. The values of $\omega_{\min}$ and $\omega_{\max}$ were chosen for LDIW-PSO based on the experiments conducted in [9]; values for c_1 and c_2 were chosen for RIW-PSO based on the recommendation in [23] and it was also used for LDIW-PSO because it was discovered in course of the experiments in this paper that these values make LDIW-PSO perform better than

Table 2: Benchmark problems.

Number	Problem	Dimensions	Optimal value	Success threshold
1	Ackley	10, 20, 30	0	10^{-5}
2	Booth	2	0	10^{-5}
3	Easom	2	−1	−1
4	Griewank	10, 20, 30	0	10^{-5}
5	Dixon-Price	10, 20, 30	0	10^{-5}
6	Levy	10, 20, 30	0	10^{-5}
7	Michalewicz	5	−4.687	−4.687
8	Noisy Quartic	10, 20, 30	0	10^{-5}
9	Noncontinous Rastrigin	10, 20, 30	0	20
10	Rastrigin	10, 20, 30	0	20
11	Rosenbrock	10, 20, 30	0	20
12	Rotated Ellipsoid	10, 20, 30	0	10^{-5}
13	Salomon	5	0	10^{-5}
14	Schaffer's f6	2	0	10^{-5}
15	Schwefel	10, 20, 30		
16	Schwefel P2.22	10, 20, 30	0	10^{-5}
17	Shubert	2	−186.7309	−186.7309
18	Sphere	10, 20, 30	0	10^{-5}
19	Step	10, 20, 30	0	10^{-5}
20	Sum Squares	10, 20, 30	0	10^{-5}
21	Trid	6	−50	−50

the commonly used value of 2.0. The settings for $V_{\min}$ and $V_{\max}$ were done based on the outcome of experimental studies in [8].

5.3. Performance Measurement. The efficiency of the algorithms was tested against the set of benchmark problems given in Table 2 and numerical results obtained were analyzed using the criteria that are listed below. All the results are presented in Tables 4 to 20.

(i) Best fitness solution: the best of the fitness solution among the solutions obtained during the runs.

(ii) Mean best fitness solution: this is a measure of the precision (quality) of the result that the algorithm can get within given iterations in all the 25 runs.

(iii) Standard deviation (Std. Dev.) of mean best fitness solution over 25 runs: this measures the algorithm's stability and robustness.

(iv) Average number of iterations an algorithm was able to reach the success threshold.

(v) Success rate (SR) = (Number of successful runs/Total number of runs) × 100: this is the rate at which the success threshold is met during the independent number of runs and is a reflection of the global search ability and robustness of the algorithm.

Statistical analysis using the Wilcoxon signed rank nonparametric test with 0.05 level of significance [28, 29] was also performed using the numerical results obtained by the algorithms, while box plots were used to analyze their variability in obtaining fitness values in all the runs.

5.4. Results and Discussions. Results obtained from all the experiments are discussed in this subsection to show the overall performance of the various algorithms. Presented in Tables 4, 5, 6, 7, 8, and 9 are the numerical results obtained and used to compare R-PSO$_{\text{CLUS}}$ and L-PSO$_{\text{CLUS}}$ with GLSPSO. R-PSO$_{\text{CLUS}}$ and L-PSO$_{\text{CLUS}}$ were also compared with PSOlis using the results presented in Table 10. The results in Tables 11–18 were obtained for the scaled and nonscaled test problems listed in Table 3; the results were used to validate RIW-PSO, R-PSO$_{\text{CLUS}}$, LDIW-PSO, and L-PSO$_{\text{CLUS}}$. In each of the tables, for ease of observation, bold values represent the better results and "–" means that the algorithm could not satisfy the success threshold in any of the runs. The Wilcoxon sign rank nonparametric test, which is used as an alternative to the paired t-test when the results cannot be assumed to be normally distributed, was applied to test the statistical significance differences between RIW-PSO and R-PSO$_{\text{CLUS}}$ as well as LDIW-PSO and L-PSO$_{\text{CLUS}}$.

5.4.1. Comparison of R-PSO$_{CLUS}$ and Golden Ratio Local Search Based PSO (GLSPSO). The results in Tables 4–6 show the performance and abilities of R-PSO$_{\text{CLUS}}$ and GLSPSO optimizing the test problems over three different problem dimensions. The results of GLSPSO were obtained from [15]. A large problem space was used for all the problems to verify the superiority between the two different local search techniques hybridized with the PSO variants. From the results it is evident that R-PSO$_{\text{CLUS}}$ is superior to GLSPSO. Apart from *Ackley* problem (across the three dimensions) and *Rosenbrock* (in dimension 100), R-PSO$_{\text{CLUS}}$ outperformed GLSPSO. It was able to obtain optimal minimum for some of the problems, demonstrating better exploitation ability, convergence precision, and solution quality.

5.4.2. Comparison between L-PSO$_{CLUS}$ and GLSPSO. To further demonstrate the efficiency of the proposed local search technique, L-PSO$_{\text{CLUS}}$ was also implemented and results were compared with the results of GLSPSO obtained from [15]. Three different types of dimensions were also used for the problems. As can be observed in Tables 7–9, across the three dimensions, GLSPSO was only able to perform better than L-PSO$_{\text{CLUS}}$ in *Ackley* problem. Apart from *Griewank* problem (dimension 10), GLSPSO was outperformed by L-PSO$_{\text{CLUS}}$ in the remaining four problems. L-PSO$_{\text{CLUS}}$ was able to obtain global optimum for *Griewank* and *Sphere* problems across the three dimensions, but, for *Rastrigin*, it was able to get global minimum for dimension 10. Again, the proposed local search technique demonstrates better exploitation ability than GLSPSO.

5.4.3. Comparison of R-PSO$_{CLUS}$ and L-PSO$_{CLUS}$ with PSOlis. Presented in Table 10 is the result obtained by L-PSO$_{\text{CLUS}}$

Table 3: Benchmark problems.

Number	Problem	Formulation	Feature	Search range
1	Ackley	$f(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	MN	±32
2	Booth	$f(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	MN	±10
3	Easom	$f(\vec{x}) = -\cos(x_1)\cos(x_2)\exp\left(-(x_1-\pi)^2 - (x_2-\pi)^2\right)$	UN	±100
4	Griewank	$f(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d}x_i^2\right) - \left(\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$	MN	±600
5	Dixon-Price	$f(\vec{x}) = (x_1 - 1)^2 + \sum_{i=2}^{d} i\left(2x_i^2 - x_{i-1}\right)^2$	UN	±10
6	Levy	$f(\vec{x}) = \sin^2(\pi y_1) + \sum_{i=1}^{d-1}(y_i - 1)^2\left(1 + 10\sin^2(\pi y_i + 1)\right) + (y_d - 1)^2\left(1 + \sin^2(2\pi x_d)\right)$, where $y_i = 1 + \frac{x_i - 1}{4}$, and $i = 1, 2, \ldots, d$	MN	±10
7	Michalewicz	$f(\vec{x}) = -\sum_{i=1}^{d}\sin(x_i)\left[\sin\left(\frac{ix_i^2}{\pi}\right)\right]^{2m}$, where $m = 10$	MS	$[0, \pi]$
8	Noisy Quartic	$f(\vec{x}) = \sum_{i=1}^{d} ix_i^4 + \text{random}(0, 1)$	US	±1.28
9	Noncontinous Rastrigin	$f(\vec{x}) = \sum_{i=1}^{d}\left(y_i^2 - 10\cos(2\pi y_i) + 10\right)$, $y_i = \begin{cases} x_i & \text{if } \lvert x_i\rvert < 0.5 \\ \frac{\text{round}(2x_i)}{2} & \text{if } \lvert x_i\rvert \geq 0.5 \end{cases}$	MS	±5.12
10	Rastrigin	$f(\vec{x}) = \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	MS	±5.12
11	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{d-1}\left(100\left(x_{i+1} - x_i^2\right)^2\right) + (x_i - 1)^2$	UN	±30
12	Rotated Ellipsoid	$f(\vec{x}) = \sum_{i=1}^{d}\left(\sum_{j=1}^{i}x_j\right)^2$	UN	±100
13	Salomon	$f(\vec{x}) = -\cos\left(2\pi\sum_{i=1}^{d}x_i^2\right) + 0.1\sqrt{\sum_{i=1}^{d}x_i^2} + 1$	MN	±100
14	Schaffer's f6	$f(\vec{x}) = \sum_{i=1}^{d-1}\left(0.5 + \frac{\sin^2\left(\sqrt{x_{i+1}^2 + x_i^2}\right) - 0.5}{\left(0.001\left(x_{i+1}^2 + x_i^2\right) + 1\right)^2}\right)$	MN	±100
15	Schwefel	$f(\vec{x}) = \sum_{i=1}^{n} - x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	MS	±500
16	Schwefel P2.22	$f(\vec{x}) = \sum_{i=1}^{d}\lvert x_i\rvert + \prod_{i=1}^{d}\lvert x_i\rvert$	UN	±10
17	Shubert	$f(\vec{x}) = \prod_{i=1}^{d}\left(\sum_{j=1}^{5} j\cos\left((j+1)sx_i + j\right)\right)$	MN	±10
18	Sphere	$f(\vec{x}) = \sum_{i=1}^{d}x_i^2$	US	±100
19	Step	$f(\vec{x}) = \sum_{i=1}^{d}\left(\lfloor x_i + 0.5\rfloor\right)^2$	US	±10
20	SumSquares		US	±10
21	Trid	$f(\vec{x}) = \sum_{i=1}^{d}(x_i - 1)^2 - \sum_{i=2}^{d}x_i x_{i-1}$	UN	$\pm d^2$

Table 4: Comparison between GLSPSO and R-PSO$_{CLUS}$ for problems with dimension of 10.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$
Best fitness	0.0364	**0.0000**	4.2879e − 04	**0.0000e + 00**	8.8062	**0.0000**	2.6188	**0.0000**	4.7832e − 04	**3.1461e − 43**
Mean fitness	**0.3413**	17.1371	0.0041	**0.0016**	29.4936	**0.0000**	9.0025	**1.9971**	0.0142	**0.0000**
Worst fitness	**1.2653**	20.0888	**0.0419**	0.0791	50.4781	**0.0000**	18.9887	**3.1444**	0.0476	**0.0000**
Std. Dev.	**0.2762**	6.7543	**0.0061**	0.0111	10.4372	**0.0000**	0.034	**0.7262**	0.0123	**0.0000**

Table 5: Comparison between GLSPSO and R-PSO$_{CLUS}$ for problems with dimension of 30.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$
Best fitness	**2.2784**	20.3075	0.0897	**0.0000**	109.5946	**13.9247**	175.8785	**22.7589**	1.9123	**0.0000**
Mean fitness	**2.8398**	20.4778	0.1257	**0.0000**	185.5221	**36.3715**	218.4976	**27.5147**	2.7449	**0.0000**
Worst fitness	**3.2952**	20.5792	0.2074	**0.0000**	229.6229	**72.6581**	259.2466	**76.7433**	3.9559	**0.0000**
Std. Dev.	0.2273	**0.0574**	0.0274	**0.0000**	24.9829	**16.4882**	21.8027	**9.9182**	0.4840	**0.0000**

Table 6: Comparison between GLSPSO and R-PSO$_{CLUS}$ for problems with dimension of 100.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$	GLSPSO	R-PSO$_{CLUS}$
Best fitness	**3.5148**	20.9666	0.3195	**0.0022**	792.004	**293.5795**	**1378.0**	1867.2669	23.0614	**0.1970**
Mean fitness	**3.6709**	21.0691	0.4242	**0.0230**	881.0822	**688.0048**	**1602.0**	24909.8486	27.2534	**4.7232**
Worst fitness	**3.7664**	21.1306	0.4992	**0.0923**	934.9773	**848.9927**	**1763.0**	95519.4585	29.1615	**16.1174**
Std. Dev.	0.0551	**0.0316**	0.0303	**0.0255**	**35.2341**	103.1854	**90.2874**	21083.5791	**1.2253**	4.2498

Table 7: Comparison between GLSPSO and L-PSO$_{CLUS}$ for problems with dimension of 10.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$
Best fitness	0.0364	**0.0000**	4.2879e − 04	**0.0000e + 00**	8.8062	**0.0000**	2.6188	**0.0000**	4.7832e − 04	**6.4151e − 76**
Mean fitness	**0.3413**	18.2504	**0.0041**	0.0042	29.4936	**0.0000**	9.0025	**1.0516**	0.0142	**0.0000**
Worst fitness	**1.2653**	20.0771	**0.0419**	0.1008	50.4781	**0.0000**	18.9887	**2.8033**	0.0476	**0.0000**
Std. Dev.	**0.2762**	5.4640	**0.0061**	0.0186	10.4372	**0.0000**	**0.034**	0.6449	0.0123	**0.0000**

Table 8: Comparison between GLSPSO and L-PSO$_{CLUS}$ for problems with dimension of 30.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$	GLSPSO	L-PSO$_{CLUS}$
Best fitness	**2.2784**	20.3184	0.0897	**0.0000**	109.5946	**0.1444**	175.8785	**0.0000**	1.9123	**0.0000**
Mean fitness	**2.8398**	20.4631	0.1257	**0.0000**	185.5221	**18.7372**	218.4976	**25.1359**	2.7449	**0.0000**
Worst fitness	**3.2952**	20.5734	0.2074	**0.0000**	229.6229	**38.8433**	259.2466	**77.4444**	3.9559	**0.0000**
Std. Dev.	0.2273	**0.0615**	0.0274	**0.0000**	24.9829	**8.4570**	21.8027	**13.2536**	0.4840	**0.0000**

Table 9: Comparison between GLSPSO and L-PSO$_{\text{CLUS}}$ for problems with dimension of 100.

Problem	Ackley		Griewank		Rastrigin		Rosenbrock		Sphere	
Algorithm	GLSPSO	L-PSO$_{\text{CLUS}}$	GLSPSO	L-PSO$_{\text{CLUS}}$	GLSPSO	L-PSO$_{\text{CLUS}}$	GLSPSO	L-PSO$_{\text{CLUS}}$	GLSPSO	L-PSO$_{\text{CLUS}}$
Best fitness	**3.5148**	20.2136	0.3195	**0.0000**	792.004	**212.0416**	1378.0	**93.7390**	23.0614	**0.0000**
Mean fitness	**3.6709**	21.0491	0.4242	**0.0000**	881.0822	**366.6521**	1602.0	**107.2300**	27.2534	**0.0000**
Worst fitness	**3.7664**	21.1152	0.4992	**0.0000**	934.9773	**504.2204**	1763.0	**428.1758**	29.1615	**0.0000**
Std. Dev.	**0.0551**	0.1254	0.0303	**0.0000**	**35.2341**	68.2009	90.2874	**56.9231**	1.2253	**0.0000**

Table 10: Comparison between PSOlis, R-PSO$_{\text{CLUS}}$ and L-PSO$_{\text{CLUS}}$.

Problem	Algorithm		
	PSOlis	R-PSO$_{\text{CLUS}}$	L-PSO$_{\text{CLUS}}$
Ackley	$4.081e-03$	**9.263e − 13**	**3.7135e − 15**
Griewank	$2.673e-02$	**6.921e − 03**	**2.945e − 07**
Rastrigin	2.005	**1.948e − 09**	**8.893e − 06**
Rosenbrock	3.987	**5.180e − 01**	**2.338e − 01**
Sphere	$6.137e-14$	**2.197e − 27**	**2.401e − 54**

in comparison with result for PSOlis from [18]. Again, the outstanding performance of L-PSO$_{\text{CLUS}}$ over its competitor is evidence that the proposed local search technique is very efficient and capable of complementing the global search ability of PSO to obtain quality results by making it overcome premature convergence.

5.4.4. Comparison between RIW-PSO and R-PSO$_{\text{CLUS}}$. The results presented in Table 11 are for the nonscaled test problems as optimized by the two algorithms while those in Tables 12–14 are for the scaled problems with 10, 20, and 30 dimensions, respectively. In Table 19 are the results obtained using the Wilcoxon sign rank nonparametric test.

(1) Results for the Nonscaled Problems. For the 7 nonscaled problems, Table 11 shows that there are no performance differences between the two algorithms in optimizing *Booth*, *Easom*, *Shubert,* and *Trid* problems. For *Michalewicz*, *Schaffer's f6*, and *Salomon*, R-PSO$_{\text{CLUS}}$ obtained more quality solutions and demonstrated better global search ability than RIW-PSO. The convergence curves in Figures 1(c) and 1(d) show that R-PSO$_{\text{CLUS}}$ has faster and better convergence. However, the P value (0.190) obtained from the Wilcoxon sign test shown in Table 19 revealed that there is no statistical difference in the performance between the two algorithms for the nonscaled problems. Also, the two algorithms have equal median fitness.

(2) Results for 10-Dimensional Problems. For the scaled problems with 10 dimensions, Table 12 clearly reveals great differences in performance between RIW-PSO and R-PSO$_{\text{CLUS}}$. The two algorithms successfully optimized *Rastrigin*, *Rosenbrock*, *Rotated Ellipsoid*, *Schwefel 2.22*, *Sphere,* and *Sum Squares* problems with equal success rate of 100%, but R-PSO$_{\text{CLUS}}$ obtained significantly better mean fitness and standard deviation with fewer number of iterations. R-PSO$_{\text{CLUS}}$ was able to obtain the minimum optima for both *Rastrigin* and *Step* problems. For the other problems, R-PSO$_{\text{CLUS}}$ clearly outperformed RIW-PSO in solution quality, convergence precision, global search ability, and robustness, though none of them could meet the success threshold in optimizing the *Noisy Quartic* problem. The P value (0.001) obtained from the Wilcoxon sign test presented in Table 19 indicates that there is statistically significant difference in performance between the two algorithms with a large effect size of $r = 0.6$ in favour of R-PSO$_{\text{CLUS}}$. The median fitness is also an evidence of this.

(3) Results for 20-Dimensional Problems. The same set of experiments was performed using the same scaled problems but with their dimensions increased to 20, which also increased their complexities except *Griewank*. The numerical results in Table 13 also show that there are great differences in performance between RIW-PSO and R-PSO$_{\text{CLUS}}$. The two algorithms had equal success rate of 100% in optimizing *Rosenbrock*, *Schwefel 2.22*, *Sphere,* and *Sum Squares* problems with R-PSO$_{\text{CLUS}}$ obtaining significantly better mean fitness (except *Rosenbrock*), standard deviation, and fewer number of iterations. Out of the remaining 10 problems R-PSO$_{\text{CLUS}}$ outperformed RIW-PSO in 9 of them with better solution quality, convergence precision, global search ability, and robustness; it also had success rate of 100% in 6 of the problems compared with RIW-PSO and was able to obtain global minimum for *Griewank* and *Step* problems. The algorithms could not meet the success criteria in optimizing the *Dixon-Price*, *Noisy Quartic,* and *Schwefel* problems, but R-PSO$_{\text{CLUS}}$ still performed better than RIW-PSO. The P value (0.023) from Wilcoxon sign test as shown in Table 19 also indicates that there is statistically significant difference in performance between the two algorithms with a medium effect size of $r = 0.43$ in favour of R-PSO$_{\text{CLUS}}$. The median fitness value of R-PSO$_{\text{CLUS}}$ is smaller than that of RIW-PSO.

(4) Results for 30-Dimensional Problems. Table 14 represents the experimental results obtained by the two algorithms using the same scaled problems but with their dimensions scaled to 30, which further increased their complexities except *Griewank*. The results further show the great differences in performance between RIW-PSO and R-PSO$_{\text{CLUS}}$. Out of the 14 problems R-PSO$_{\text{CLUS}}$ had 100% success rate in 7 of them (4 multimodal and 3 unimodal) while RIW-PSO could only have in 3 of them (all unimodal). The two algorithms

Table 11: Results for RIW-PSO and R-PSO$_{\text{CLUS}}$ for the 7 nonscaled benchmark problems.

Problem	Booth		Easom		Michalewicz		Schaffer's f6		Salomon		Shubert		Trid-6	
Algorithm	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$
Best fitness	0.0000e + 00	0.0000e + 00	−1.0000e + 00	−1.0000e + 00	−3.3453e + 00	**−4.4371e** + **00**	0.0000e + 00	0.0000e + 00	9.9833e − 02	**7.3608e** − **29**	−1.8673e + 02	−1.8673e + 02	−5.0000e + 00	−5.0000e + 00
Mean fitness	0.0000e + 00	0.0000e + 00	−1.0000e + 00	−1.0000e + 00	−2.6034e + 00	**−4.1008e** + **00**	4.7052e − 03	**1.1659e** − **03**	9.9833e − 02	**7.0073e** − **02**	−1.8673e + 02	−1.8673e + 02	−5.0000e + 00	−5.0000e + 00
Std. Dev.	0.0000e + 00	0.0000e + 00	6.6613e − 16	6.6613e − 16	4.2719e − 01	**1.7866e** − **01**	4.8183e − 03	**3.1573e** − **03**	**6.2063e** − **18**	4.4236e − 02	**3.7706e** − **14**	4.0594e − 14	8.6628e − 14	**7.0555e** − **14**
Av. iteration	39.12	**37.92**	55.0	**45.48**	—	—	133.83	**109.95**	—	**923.40**	**71.8**	107.16	114.40	**110.20**
SR (%)	100	100	100	100	0	0	48	**88**	0	**20**	100	100	100	100

Table 12: Results for RIW-PSO and R-PSO$_{CLUS}$ for the 14 scaled benchmark problems with dimension of 10.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$
Best fitness	3.9968$e-$15	**4.4409e−16**	7.3960$e-$03	**0.0000e+00**	2.4652$e-$31	2.4652$e-$31	1.4997$e-$32	1.4997$e-$32	5.9308$e-$04	**2.7756e−05**	6.0002$e+$00	**0.0000e+00**	3.9767$e+$00	**0.0000e+00**
Mean fitness	1.0316$e-$01	**2.8599e−15**	6.6238$e-$02	**2.8539e−03**	6.1333$e-$01	6.1333$e-$01	4.1630$e-$01	**1.4997e−32**	4.6455$e-$03	**1.1590e−04**	1.1640$e+$01	**7.1054e−17**	1.2208$e+$01	**0.0000e+00**
Std. Dev.	5.0537$e-$01	**1.6573e−15**	3.3218$e-$02	**1.3981e−02**	1.8086$e-$01	1.8086$e-$01	7.5307$e-$01	**0.0000e+00**	3.2781$e-$03	**5.9530e−05**	4.3533$e+$00	**3.4809e−16**	4.4684$e+$00	**0.0000e+00**
Av. iteration	287.88	**263.68**	—	**464.16**	295.50	**258.50**	127.31	**99.32**	—	—	40.49	**12.44**	49.44	**25.00**
SR (%)	96	**100**	0	**96**	8	8	52	**100**	0	0	92	**100**	100	100
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$
Best fitness	**4.6541e−03**	3.7243$e-$01	2.5690$e-$27	**5.9210e−29**	−3.2818$e+$03	**−4.1898e+03**	1.8348$e-$32	**2.7555e−34**	7.3673$e-$61	**5.4496e−61**	0.0000$e+$00	0.0000$e+$00	**8.1835e−62**	1.9089$e-$61
Mean fitness	1.4459$e+$00	**7.1832e−01**	1.0457$e-$21	**3.5437e−24**	−2.6199$e+$03	**−4.1384e+03**	3.3382$e-$28	**1.2830e−30**	4.1760$e-$53	**9.2787e−55**	8.0000$e-$01	**0.0000e+00**	2.1909$e-$53	**9.3329e−54**
Std. Dev.	1.6362$e+$00	**2.3127e−01**	3.0984$e-$21	**1.1617e−23**	3.3350$e+$02	**9.7987e+01**	1.3406$e-$27	**4.6701e−30**	1.2467$e-$52	**3.6764−54**	1.9183 + 00	**0.0000e+00**	1.0052$e-$52	**3.3839e−53**
Av. iteration	89.28	**35.16**	426.60	**337.16**	—	**877.69**	268.84	**222.04**	180.12	**158.08**	88.72	**15.68**	143.52	**123.48**
SR (%)	100	100	100	100	0	**52**	100	100	100	100	72	**100**	100	100

Table 13: Results for RIW-PSO and R-PSO$_{\text{CLUS}}$ for the 14 scaled benchmark problems with dimension of 20.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$
Best fitness	7.5495$e-15$	**3.9968e−15**	0.0000$e+00$	**0.0000e+00**	6.6667$e-01$	6.6667$e-01$	1.4997$e-32$	1.4997$e-32$	3.9758$e-03$	**8.2784e−05**	1.2000$e+01$	**1.6698e−03**	1.4000$e+01$	**9.8524e−03**
Mean fitness	2.6017$e-01$	**3.9968e−15**	2.7935$e-02$	**0.0000e+00**	6.6667$e-01$	6.6667$e-01$	7.8421$e-01$	**1.4997e−32**	1.0286$e-02$	**2.7771e−04**	3.0521$e+01$	**4.4448e+00**	2.8521$e+01$	**4.3599e+00**
Std. Dev.	6.0386$e-01$	**0.0000e+00**	1.8216$e-02$	**0.0000e+00**	**2.4222e−16**	4.1352$e-08$	8.0230$e-01$	**0.0000e+00**	5.0796$e-03$	**1.3566e−04**	1.0922$e+01$	**2.5668e+00**	9.5087$e+00$	**1.9684e+00**
Av. iteration	580.67	**488.52**	402.00	**382.80**	—	—	248.00	**200.04**	—	—	51.5	**86.72**	56.2	89.08
SR (%)	84	**100**	4	**100**	0	0	16	**100**	0	0	16	**100**	20	**100**
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$	RIW-PSO	R-PSO$_{\text{CLUS}}$
Best fitness	**8.2827e−01**	7.4672$e+00$	8.7694$e-04$	**7.9922e−09**	−5.9318$e+03$	**−8.1903e+03**	2.4775$e-22$	**3.5223e−26**	3.1840$e-43$	**2.9531e−44**	0.0000$e+00$	**0.0000e+00**	4.8722$e-45$	**2.1277e−46**
Mean fitness	**9.1608e+00**	1.0488$e+01$	**7.2146e−06**	1.9591$e-05$	−4.5638$e+03$	**−7.0464e+03**	8.5083$e-17$	**4.9420e−23**	3.9759$e-37$	**8.4151e−38**	1.5600$e+00$	**0.0000e+00**	8.3393$e-39$	**1.4859e−39**
Std. Dev.	3.5710$e+00$	**8.1620e−01**	**1.9200e−05**	7.2466$e-05$	6.0867$e+02$	**8.4004e+02**	3.9686$e-16$	**1.2554e−22**	1.2218$e-36$	**3.0436e−37**	1.6020$e+00$	**0.0000e+00**	1.4280$e-38$	**4.9918e−39**
Av. iteration	263.40	**133.72**	1763.10	**1636.74**	—	—	575.92	**418.88**	354.00	**309.04**	646.57	**19.36**	310.72	**268.12**
SR (%)	100	100	84	**92**	0	0	100	100	100	100	28	**100**	100	100

Table 14: Results for RIW-PSO and R-PSO$_{CLUS}$ for the 14 scaled benchmark problems with dimension of 30.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$
Best fitness	1.4655$e-$14	**3.9968e−15**	0.0000$e+$00	**0.0000e+00**	6.6667$e-$01	6.6667$e-$01	2.6858$e-$01	**1.4997e−32**	3.3454$e-$03	**1.0873e−04**	1.9001$e+$01	**5.7746e−02**	1.7895$e+$01	**0.0000e+00**
Mean fitness	5.3345$e-$01	**4.1389e−15**	1.1200$e-$02	**0.0000e+00**	6.6667$e-$01	6.6716$e-$01	1.8286$e+$00	**1.4997e−32**	1.1252$e-$02	**3.1438e−04**	4.3331$e+$01	**9.0806e+00**	3.2489$e+$01	**6.5810e+00**
Std. Dev.	8.1543$e-$01	**6.9619e−16**	1.4757$e-$02	**0.0000e+00**	**3.2634e−16**	2.4146$e-$03	1.1781$e+$00	**0.0000e+00**	4.3926$e-$03	**1.1754e−04**	1.8889$e+$01	**5.1000e+00**	1.0662$e+$01	**5.7209e+00**
Av. iteration	959.88	**776.20**	628.00	**525.48**	—	—	—	**348.76**	—	—	107	**377.72**	141.00	**379.35**
SR (%)	64	**100**	36	**100**	0	0	0	**100**	0	0	4	**100**	4	**92**
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$	RIW-PSO	R-PSO$_{CLUS}$
Best fitness	**4.0555e−02**	1.8942$e+$01	3.2599$e-$04	**1.7148e−04**	−7.9654$e+$03	**−1.0692e+04**	2.7819$e-$17	**1.0011e−24**	5.7947$e-$39	**5.0490e−42**	0.0000$e+$00	0.0000$e+$00	1.7310$e-$36	**1.2387e−44**
Mean fitness	2.3794$e+$01	**1.9930e+01**	**5.5035e−03**	1.8327$e-$02	−6.7376$e+$03	**−9.1033e+03**	4.1597$e-$13	**6.4368e−23**	4.1277$e-$33	**3.6673e−36**	3.7600$e+$00	**0.0000e+00**	3.9240$e-$33	**5.9219e−39**
Std. Dev.	1.9052$e+$01	**5.2276e−01**	**8.1952e−03**	4.4826$e-$02	8.7747$e+$02	**8.6528e+02**	1.5547$e-$12	**7.6462e−23**	1.1473$e-$32	**8.2055e−36**	2.3200$e+$00	**0.0000e+00**	1.3909$e-$32	**1.2917e−38**
Av. iteration	1523.36	2813.14	—	—	—	—	1139.64	**658.68**	595.52	**505.80**	2881.00	**21.48**	526.28	**436.08**
SR (%)	44	**56**	0	0	0	0	100	100	100	100	4	**100**	100	100

Table 15: Results for LDIW-PSO and L-PSO$_{\text{CLUS}}$ for the 7 nonscaled benchmark problems.

Problem	Booth		Easom		Michalewicz		Schaffer's f6		Salomon		Shubert		Trid-6	
Algorithm	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$	LDIW-PSO	L-PSO$_{\text{CLUS}}$
Best fitness	0.0000e + 00	0.0000e + 00	−1.0000e + 00	−1.0000e + 00	−3.4792e + 00	**−4.3762e + 00**	0.0000e + 00	0.0000e + 00	9.9833e − 02	**3.3189e − 65**	−1.8673e + 02	−1.8673e + 02	−5.0000e + 01	−5.0000e + 01
Mean fitness	0.0000e + 00	0.0000e + 00	−1.0000e + 00	−1.0000e + 00	−2.6736e + 00	**−4.1056e + 00**	1.5545e − 03	**1.1659e − 03**	9.9833e − 02	**8.7853e − 02**	−1.8673e + 02	−1.8673e + 02	−50000e + 01	−50000e + 01
Std. Dev.	0.0000e + 00	0.0000e + 00	6.6613e − 16	6.6613e − 16	3.6921e − 01	**1.6196e − 01**	3.5619e − 03	**3.1573e − 03**	**7.3434e − 18**	3.2442e − 02	**3.9382e − 14**	7.5559e − 05	**7.8572e − 14**	9.3154e − 14
Av. iteration	**79.48**	81.16	107.16	**73.12**	—	—	279.90	**174.14**	—	**742**	**175.52**	208.20	287.48	**281.72**
SR (%)	100	100	100	100	0	0	84	**88**	0	**12**	**100**	96	100	100

Table 16: Results for LDIW-PSO and L-PSO$_{CLUS}$ for the 14 scaled benchmark problems with dimension of 10.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	3.9968$e-15$	**4.4409e−16**	2.9510$e-02$	**0.0000e+00**	6.6667$e-01$	6.6667$e-01$	1.4997$e-31$	**1.4997e−32**	3.4641 − 04	**1.1331e−05**	4.0001$e+00$	**0.0000e+00**	2.9825$e+00$	**0.0000e+00**
Mean fitness	4.9916$e-15$	**2.4335e−15**	8.4720$e-02$	**1.4639e−06**	6.6667$e-01$	6.6667$e-01$	1.9892$e-01$	**1.4997e−32**	2.9606$e-03$	**9.7778e−05**	1.2200$e+01$	**4.0007e−02**	9.9019$e+00$	**0.0000e+00**
Std. Dev.	**1.5952e−15**	1.7635$e-15$	2.7646$e-02$	**7.1705e−06**	1.5401$e-10$	**1.2278e−12**	4.4094$e-01$	**0.0000e+00**	2.0277$e-03$	**6.3873e−05**	6.2484$e+00$	**1.9600e−01**	5.2190$e+00$	**0.0000e+00**
Av. iteration	500.92	**486.84**	—	**703.83**	—	—	309.58	**281.36**	—	—	40.86	**12.88**	45.13	**26.24**
SR (%)	100	100	0	**96**	0	0	76	**100**	0	0	88	**100**	96	**100**
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	**2.7701e−03**	1.5581$e-01$	2.1928$e-44$	**1.2232e−52**	−3.2028$e+03$	**−4.1898e+03**	**3.3962e−48**	2.3988$e-46$	9.8201$e-110$	**3.2307e−112**	0.0000$e+00$	0.0000$e+00$	2.3361$e-110$	**1.6446e−112**
Mean fitness	9.3256$e-01$	**7.2655e−01**	5.8802$e-32$	**4.6946e−47**	−2.5079$e+03$	**−4.1317e+03**	8.0458$e-23$	**5.4599e−29**	3.1249$e-101$	**1.7383e−106**	1.6000$e-01$	**0.0000e+00**	**1.6238e−105**	6.7305$e-105$
Std. Dev.	1.3984$e+00$	**5.1516e−01**	2.8672$e-31$	**1.6736e−46**	4.0568$e+02$	**1.2489e+02**	3.9336$e-22$	**2.6683e−28**	1.2642$e-100$	**3.3735e−106**	7.8384$e-01$	**0.0000e+00**	**5.1760e−105**	3.1425$e-104$
Av. iteration	155.20	**69.64**	566.44	**559.20**	—	**798.45**	478.12	**450.76**	385.52	**374.28**	73.79	**15.76**	340.56	**327.24**
SR (%)	100	100	100	100	0	**44**	100	100	100	100	96	**100**	100	100

Table 17: Results for LDIW-PSO and L-PSO$_{CLUS}$ for the 14 scaled benchmark problems with dimension of 20.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	1.4655$e-$14	**3.9968e−15**	7.3960$e-$03	**0.0000e+00**	6.6667$e-$01	6.6667$e-$01	1.4997$e-$32	1.4997$e-$32	8.8724$e-$04	**8.2862e−05**	9.0003$e+$00	**5.4958e−02**	7.8533$e+$00	**0.0000e+00**
Mean fitness	2.7869$e-$13	**3.9968e−15**	3.5766$e-$02	**1.1969e−05**	6.6667$e-$01	6.6667$e-$01	9.4811$e-$01	**1.4997e−32**	6.2935$e-$03	**1.8753e−04**	2.9441$e+$01	**5.0733e+00**	2.2627$e+$01	**1.5954e+00**
Std. Dev.	1.1016$e-$12	**0.0000e+00**	2.3388$e-$02	**3.7730e−05**	**9.4022e−16**	2.6746$e-$08	1.0290$e+$00	**0.0000e+00**	2.8130$e-$03	**9.1506e−05**	1.1371$e+$01	**2.4613e+00**	7.0594$e+$00	**2.9944e+00**
Av. iteration	785.08	**773.48**	—	**813.50**	—	—	545.13	**518.48**	—	—	93	**88.40**	256.17	**144.32**
SR (%)	100	100	0	**88**	0	0	32	**100**	0	0	16	**100**	48	**100**
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	**8.5273e−01**	7.8279$e+$00	5.7451$e-$10	**3.1410e−20**	−5.5568$e+$03	**−79200e+03**	2.1114$e-$17	**4.5537e−28**	3.4892$e-$59	**4.4204e−68**	0.0000$e+$00	0.0000$e+$00	2.7411$e-$60	**3.2424e−71**
Mean fitness	**8.8496e+00**	9.1318$e+$00	6.1269$e-$05	**1.7093e−15**	−4.5047$e+$03	**−68180e+03**	3.5766$e-$09	**1.6526e−23**	1.0393$e-$45	**1.6274e−58**	8.0000$e-$02	**0.0000e+00**	1.4838$e-$48	**1.5378e−59**
Std. Dev.	3.4651$e+$00	**9.4307e−01**	1.7066$e-$04	**4.8692e−15**	**5.0603+02**	5.1886$e+$02	8.6693$e-$09	**3.1511e−23**	3.1193$e-$45	**5.5196e−58**	2.7129$e-$01	**0.0000e+00**	5.6876$e-$48	**7.4256e−59**
Av. iteration	594.96	419.12	1495.45	**1254.52**	—	—	820.56	**722.76**	639.88	**636.08**	147.52	**19.88**	595.80	**593.52**
SR (%)	100	100	88	**100**	0	0	100	100	100	100	92	**100**	100	100

Table 18: Results for LDIW-PSO and L-PSO$_{CLUS}$ for the 14 scaled benchmark problems with dimension of 30.

Problem	Ackley		Griewank		Dixon-Price		Levy		Noisy Quartic		Noncontinuous Rastrigin		Rastrigin	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	3.6993$e-13$	**3.9968e − 15**	2.2204$e-16$	**0.0000e + 00**	6.6667$e-01$	6.6667$e-01$	3.4554 − 28	**1.4997e − 32**	3.2281$e-03$	**8.0745e − 05**	2.0001$e+01$	**2.2356e + 00**	1.8889$e+01$	**0.0000e + 00**
Mean fitness	5.6071$e-11$	**3.9968e − 15**	1.2289$e-02$	**0.0000e + 00**	6.6667$e-01$	6.6667$e-01$	2.0125$e+00$	**1.4997e − 32**	7.2161$e-03$	**2.3200e − 04**	4.1361$e+01$	**1.2883e + 01**	3.3921$e+01$	**8.3217e + 00**
Std. Dev.	2.1743$e-10$	**0.0000e + 00**	1.5738$e-02$	**0.0000e + 00**	3.8233$e-12$	**3.1157e − 14**	1.6464$e+00$	**0.0000e + 00**	3.4677$e-03$	**1.0514e − 04**	1.3494$e+01$	**5.1402e + 00**	9.7583 + 00	**7.6697e + 00**
Av. iteration	1245.68	**1194.40**	1050.23	1060.44	—	—	928.00	**865.44**	—	—	—	**658.48**	518.00	718.21
SR (%)	100	100	52	**100**	0	0	4	**100**	0	0	0	**92**	4	**96**
Problem	Rosenbrock		Rotated Ellipsoid		Schwefel		Schwefel 2.22		Sphere		Step		Sum Squares	
Algorithm	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$	LDIW-PSO	L-PSO$_{CLUS}$
Best fitness	**1.4208e − 01**	1.4715$e+01$	1.4959$e-04$	**3.8064e − 11**	−7.7298$e+03$	**−1.0232e + 04**	6.6416$e-10$	**1.2902e − 24**	9.3155$e-41$	**1.4592e − 57**	0.0000$e+00$	0.0000$e+00$	1.0873$e-41$	**1.2031e − 57**
Mean fitness	1.8350$e+01$	**1.7789e + 01**	4.5745$e-02$	**1.8073e − 06**	−6.5026$e+03$	**−8.6435e + 03**	1.0569$e-04$	**2.4455e − 22**	1.2325$e-30$	**6.2108e − 47**	7.2000$e-01$	**0.0000e + 00**	1.6409$e-28$	**2.0496e − 48**
Std. Dev.	1.1240$e+01$	**1.8867e + 00**	1.3691$e-01$	**4.7189e − 06**	**7.1966e + 02**	1.0203$e+03$	4.0792$e-04$	**2.8267e − 22**	3.7363$e-30$	**3.0419e − 46**	8.7270$e-01$	**0.0000e + 00**	5.9071$e-28$	**9.3904e − 48**
Av. iteration	**1920.22**	2078.70	—	**2524.35**	—	—	1576.75	**1120.20**	1033.04	**1028.12**	408.92	**22.52**	968.72	**952.28**
SR (%)	92	92	0	**92**	0	0	80	**100**	100	100	52	**100**	100	100

Table 19: Wilcoxon signed rank test on mean fitness obtained by RIW-PSO and R-PSO$_{CLUS}$ for the test problems.

Measurement	Scaled problems			Nonscaled problems
	Dim = 10	Dim = 20	Dim = 30	
R-PSO$_{CLUS}$ < RIW-PSO	13	11	12	3
R-PSO$_{CLUS}$ > RIW-PSO	0	2	2	0
R-PSO$_{CLUS}$ = RIW-PSO	1	1	0	4
z	−3.190	−2.274	−2.606	−1.604
P value	0.001	0.023	0.009	0.190
r	0.600	0.430	0.490	—
Median				
RIW-PSO	0.847	0.144	0.272	−1.000
R-PSO$_{CLUS}$	0.000	0.000	0.000	−1.000

Table 20: Wilcoxon signed rank test on mean fitness obtained by LDIW-PSO and L-PSO$_{CLUS}$ for the test problems.

Measurement	Scaled problems			Nonscaled
	Dim = 10	Dim = 20	Dim = 30	
L-PSO$_{CLUS}$ < LDIW-PSO	12	12	13	3
L-PSO$_{CLUS}$ > LDIW-PSO	1	1	0	0
L-PSO$_{CLUS}$ = LDIW-PSO	1	1	1	4
z	−2.988	−2.552	−3.181	−1.604
P value	0.003	0.011	0.001	0.190
r	0.565	0.482	0.601	—
Median				
LDIW-PSO	0.044	0.021	0.029	−1.000
L-PSO$_{CLUS}$	0.000	0.000	0.000	−1.000

had equal success rate of 100% in optimizing *Schwefel 2.22*, *Sphere,* and *Sum Squares* problems with R-PSO$_{CLUS}$ obtaining significantly better mean fitness standard deviation and fewer number of iterations. Optimizing *Dixon-Price*, *Noisy Quartic*, *Rotated Ellipsoid,* and *Schwefel* problems, none of the algorithms could meet the success criteria, yet R-PSO$_{CLUS}$ still obtained better results than RIW-PSO. In all the 14 problems except *Rotated Ellipsoid*, R-PSO$_{CLUS}$ outperformed RIW-PSO and was able to obtain global minimum for *Griewank* and *Step* problems. The P value (0.009) from Wilcoxon sign test in Table 19 is a confirmatory evidence that there is statistically significant difference in performance between RIW-PSO and R-PSO$_{CLUS}$ with a large effect size of $r = 0.49$ in favour of R-PSO$_{CLUS}$. The median fitness value of R-PSO$_{CLUS}$ is also smaller than that of RIW-PSO. The convergence graphs of six 30-dimensional test problems shown in Figure 2 demonstrate the speed and ability of convergence of the two algorithms. From the graphs it is clear that R-PSO$_{CLUS}$ demonstrates better convergence and global search ability than RIW-PSO. Besides it also possesses better ability to get out of local optima.

5.4.5. Comparison between LDIW-PSO and L-PSO$_{CLUS}$. Presented in Table 15 are the results for the nonscaled test problems as optimized by the two algorithms while those in Tables 16–18 are for the scaled problems with 10, 20, and 30 dimensions, respectively. The statistical analysis done by applying Wilcoxon sign rank nonparametric test is presented in Table 20.

(1) Results for the Nonscaled Problems. Results in Table 15 show that there are no clear performance differences between LDIW-PSO and L-PSO$_{CLUS}$ in optimizing *Booth*, *Easom*, *Shubert,* and *Trid* problems; however, there are some not too significant differences in their average number of iterations to reach the success thresholds and standard deviation; in *Shubert*, LDIW-PSO obtained 100% success but L-PSO$_{CLUS}$ could not. Figures 1(a), 1(b), 1(e), and 1(f) show their convergence behaviour. Optimizing *Michalewicz*, *Schaffer's f6*, and *Salomon*, L-PSO$_{CLUS}$ obtained better quality solutions and has better search ability than LDIW-PSO. Also, the convergence graphs in Figures 1(c) and 1(d) show that L-PSO$_{CLUS}$ have faster and better convergence in *Schaffer's f6* and *Salomon*. The curves show that the two algorithms were trapped in local optima as shown by the flat parts of their curves and were able to escape from some of them. The P value (0.190) in Table 20, obtained from the Wilcoxon sign test, indicates that there is no statistically significant difference between the two algorithms in performance.

(2) Results for 10-Dimensional Problems. Optimizing the 10-dimensional scaled problems, L-PSO$_{CLUS}$ had 100% success in 10 of the 14 problems (4 multimodal and 6 unimodal) while LDIW-PSO had 100% success in 6 problems (1 multimodal and 5 unimodal) as shown in Table 16. It is only L-PSO$_{CLUS}$

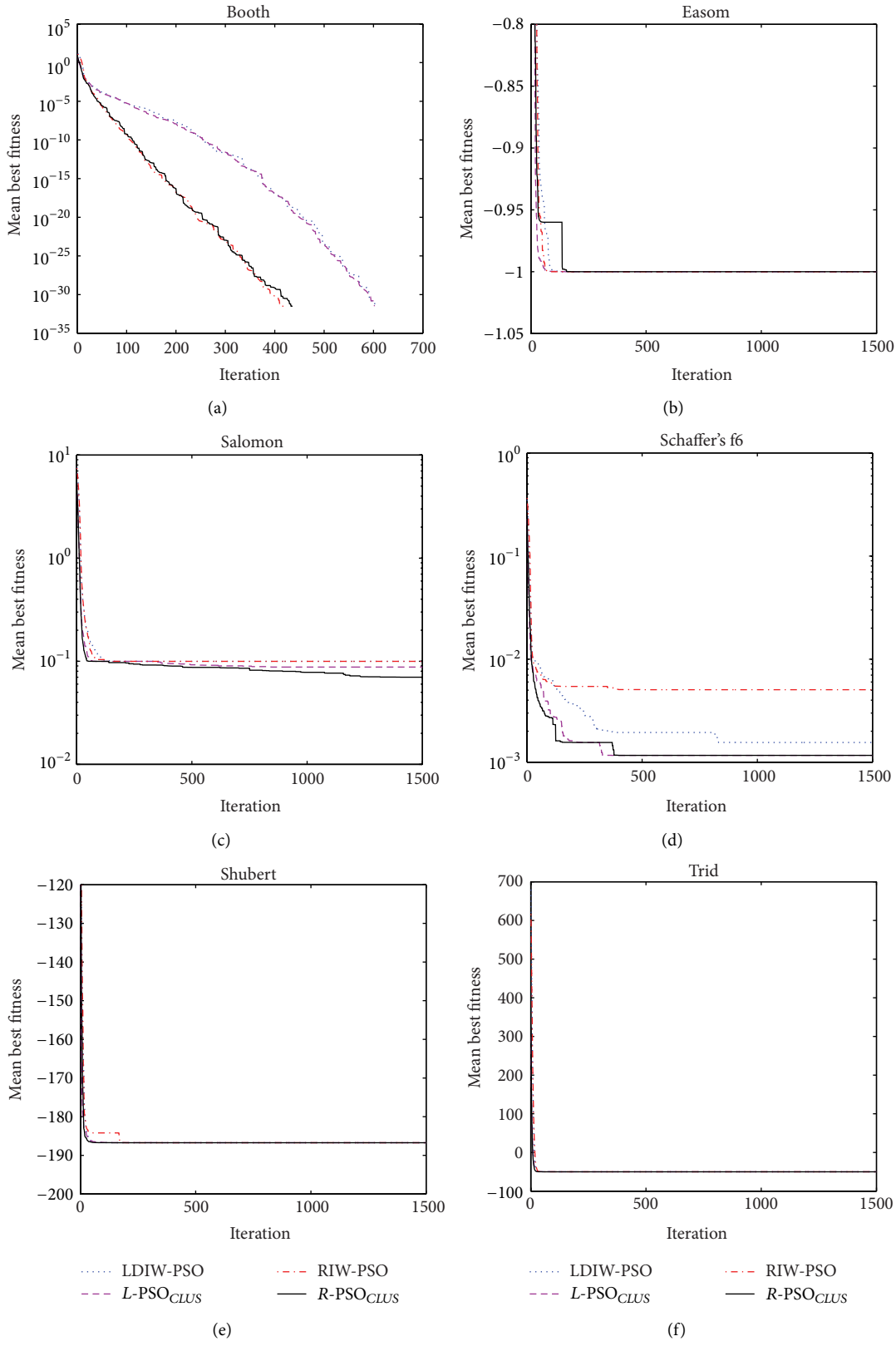


Figure 1: Convergence graphs for 6 of the nonscaled benchmark problems.

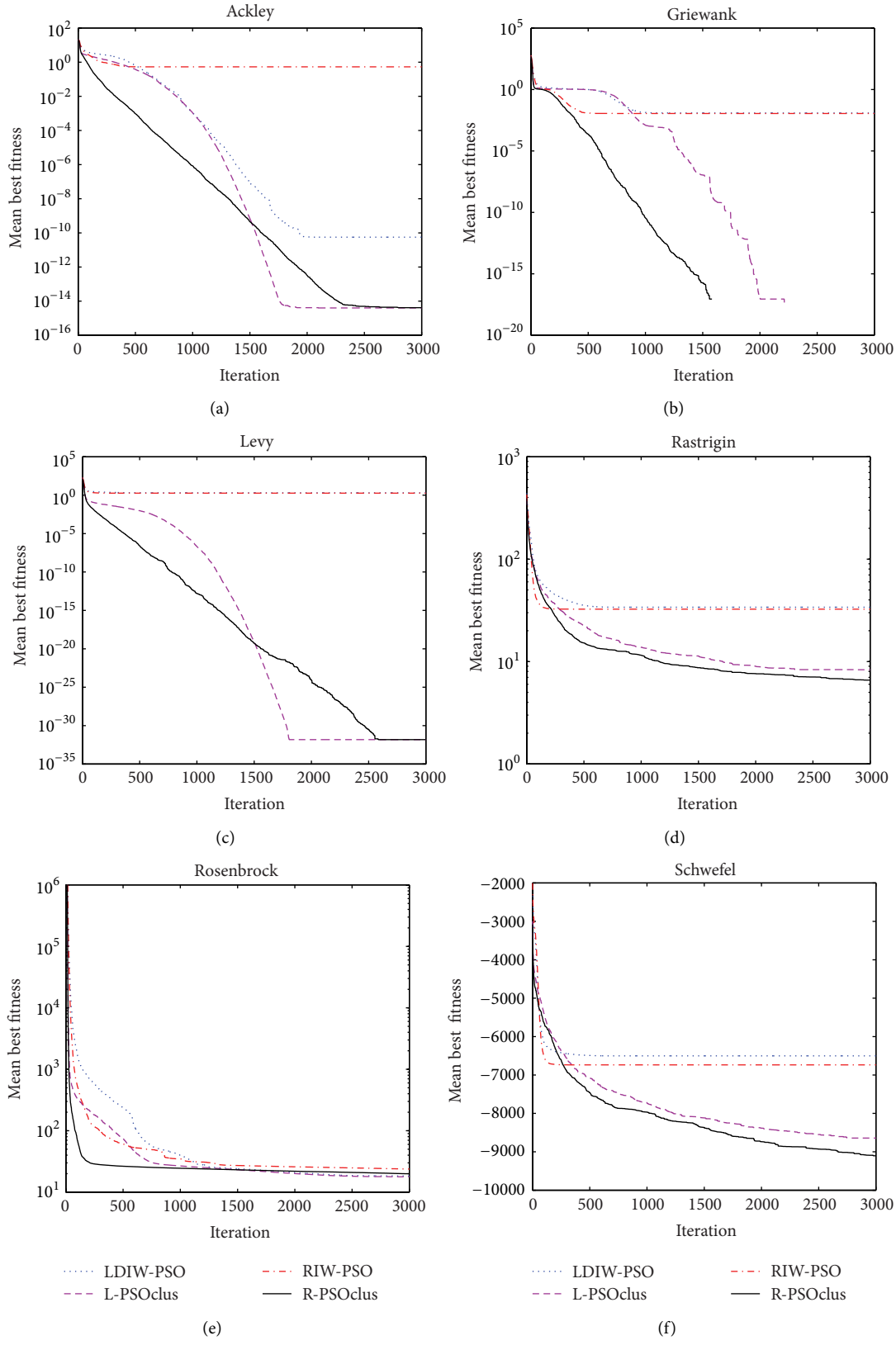


Figure 2: Convergence graphs for 6 of the scaled benchmark problems with dimension of 30.

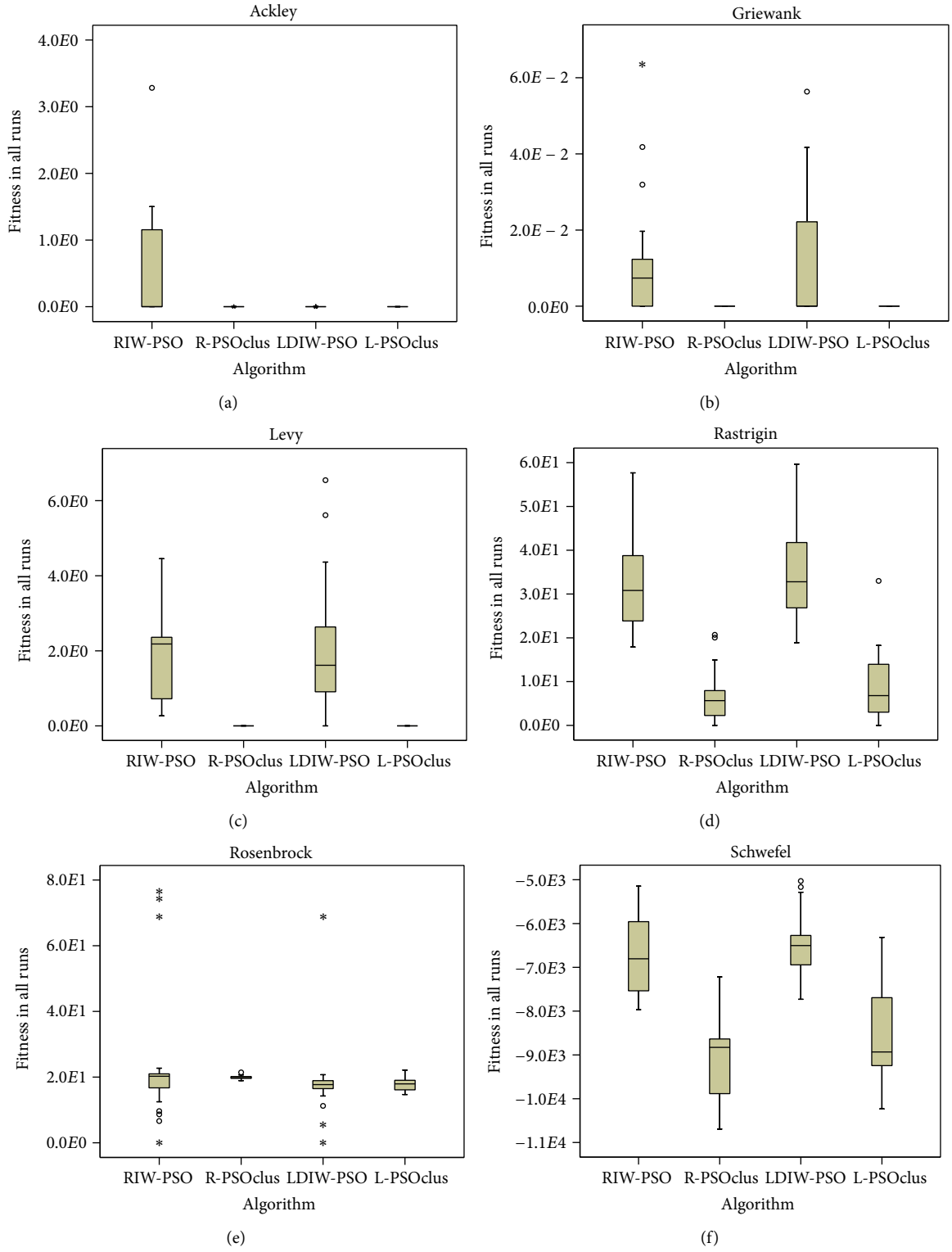


Figure 3: Box plots for 6 of the scaled test problems.

that could successfully obtain the minimum optima for both *Rastrigin* and *Step* problems but none could reach the success threshold for *Dixon-Price* and *Noisy Quartic*. In all the problems except *Dixon-Price* (where they have approximately equal performance) and *Sum Squares*, L-PSO$_{\text{CLUS}}$ clearly outperformed LDIW-PSO in obtaining better solution quality, convergence precision, global search ability, and robustness as well as fewer number of iterations. To confirm this, Wilcoxon sign test was performed on the mean best fitness over all the problems and results are presented in

Table 20; the P value (0.003) obtained indicates that there is statistically significant difference in performance between the two algorithms with a large effect size of $r = 0.565$ in favour of L-PSO$_{\text{CLUS}}$ which also has a lower median value for the mean best fitness.

(3) Results for 20-Dimensional Problems. The numerical results in Table 17 also show that there are great differences in performance between LDIW-PSO and L-PSO$_{\text{CLUS}}$ performing the same set of experiments but with the problems dimensions increased to 20. The two algorithms had equal success rate of 100% in optimizing *Ackley*, *Rosenbrock*, *Schwefel 2.22*, *Sphere,* and *Sum Squares* problems with L-PSO$_{\text{CLUS}}$ obtaining significantly better mean fitness (except *Rosenbrock*), standard deviation, and fewer number of iterations. L-PSO$_{\text{CLUS}}$ outperformed LDIW-PSO in 7 (5 multimodal and 2 unimodal) of the rest 9 problems and obtained better solution, convergence precision, global search ability, and robustness; it was also able to obtain global minimum for *Step* problem. The algorithms could not reach the success thresholds for *Dixon-Price*, *Noisy Quartic,* and *Schwefel* problems. The nonparametric test that was performed using Wilcoxon sign test, with results shown in Table 20, also confirms statistically significant difference in performance between the two algorithms with P value (0.011) and a large effect size of $r = 0.482$ in the direction of L-PSO$_{\text{CLUS}}$. The median fitness value of L-PSO$_{\text{CLUS}}$ is also smaller than that of LDIW-PSO.

(4) Results for 30-Dimensional Problems. Scaling the dimensions of test problems to 30 to further increase their complexities, except *Griewank* which decreases in complexity with increased dimension, did not affect the better performance of L-PSO$_{\text{CLUS}}$ over LDIW-PSO. Presented in Table 18 are the experimental results obtained by the two algorithms optimizing the same scaled problems. The results indicate that there are great differences between LDIW-PSO and L-PSO$_{\text{CLUS}}$ in performance. Out of the 14 problems L-PSO$_{\text{CLUS}}$ had 100% success rate in 6 of them (3 multimodal and 3 unimodal) while LDIW-PSO could only have in 3 (1 multimodal and 2 unimodal). They had equal success rate of 100% in optimizing *Ackley*, *Sphere,* and *Sum Squares* problems and 92% in *Rosenbrock* with L-PSO$_{\text{CLUS}}$ obtaining significantly better results. Optimizing *Dixon-Price*, *Noisy Quartic,* and *Schwefel* problems, none of the algorithms could reach the success threshold, yet L-PSO$_{\text{CLUS}}$ still obtained better results than LDIW-PSO, except in *Dixon-Price* where they had approximately the same performance. LDIW-PSO was not able to reach success threshold for *Noncontinuous Rastrigin* and *Rotated Ellipsoid* problems unlike L-PSO$_{\text{CLUS}}$. In all the 14 problems L-PSO$_{\text{CLUS}}$ conceded in none to LDIW-PSO and it was able to obtain global minimum for *Griewank* and *Step* problems. The P value (0.001) in Table 20 further confirms that there is statistically significant difference between LDIW-PSO and L-PSO$_{\text{CLUS}}$ with a large effect size of $r = 0.601$ in the direction of L-PSO$_{\text{CLUS}}$. The median value for the mean fitness of L-PSO$_{\text{CLUS}}$ is also smaller than that of RIW-PSO. Figure 1 shows the convergence graphs of the two algorithms. From the graphs it is clear that L-PSO$_{\text{CLUS}}$ demonstrates better convergence speed, better ability to escape premature convergence, and global search ability than LDIW-PSO.

5.4.6. Box Plots Analysis. Other than using statistical test to observe the performance of RIW-PSO, R-PSO$_{\text{CLUS}}$, LDIW-PSO, and L-PSO$_{\text{CLUS}}$, box plots analysis was also performed for 6 of the scaled test problems with 30 dimensions; the results are presented in Figures 3(a)–3(f). Box plots give a direct visual comparison of both location and the dispersion of data. The four algorithms are plotted together to optimize space. In each of the plot, RIW-PSO is compared with R-PSO$_{\text{CLUS}}$ while LDIW-PSO is compared with L-PSO$_{\text{CLUS}}$. The plots strengthen and justify the better performance of PSO when used with the proposed local search technique.

6. Conclusion

A new local search technique has been proposed in this paper with the goal of addressing the problem of premature convergence associated with particle swarm optimization algorithms. The proposed local search was used to efficiently improve the performance of two existing PSO variants, RIW-PSO and LDIW-PSO. These variants have been known to be less efficient optimizing continuous optimization problems. In this paper they were hybridized with the local search to form two other variants R-PSO$_{\text{CLUS}}$ and L-PSO$_{\text{CLUS}}$. Some well-studied benchmark problems with low and high dimensions were used to extensively validate the performance of these new variants and comparisons were made with RIW-PSO and LDIW-PSO. They were also compared with two other PSO variants in the literature, which are hybridized with different local search techniques. The experimental results obtained show that the proposed variants successfully obtain better results with high quality while demonstrating better convergence velocity and precision, stability, robustness, and global-local search ability than the competing variants. This therefore shows that the local search technique proposed can help PSO algorithms execute effective exploitation in the search space to obtain high quality results for complex continuous optimization problems. This local search technique can be used with any population-based optimization algorithms to obtain quality solutions to simple and complex optimization problem.

Further study is needed on the parameter tuning of the proposed local search technique. Empirical investigation of the behaviour of the technique in optimizing problems with noise needs further study. The scalability of the algorithms for problems with higher dimension greater than 100 is essential. Finally, the proposed algorithm can be applied to real-world optimization problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors acknowledge College of Agriculture, Engineering and Sciences, University of Kwazulu-Natal, South Africa, for their support towards this work.

References

[1] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science (MHS '95)*, pp. 39–43, Nagoya, Japan, October 1995.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.

[3] V. N. Dieu, P. Schegner, and W. Ongsakul, "A newly improved particle swarm optimization for economic dispatch with valve point loading effects," in *Proceedings of the IEEE Power and Energy Society General Meeting*, pp. 1–8, July 2011.

[4] J. Bai, X. Zhang, and Y. Guo, "Different inertia weight PSO algorithm optimizing SVM kernel parameters applied in a speech recognition system," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA '09)*, pp. 4754–4759, August 2009.

[5] M. M. Mansour, S. F. Mekhamer, and N. E.-S. El-Kharbawe, "A modified particle swarm optimizer for the coordination of directional overcurrent relays," *IEEE Transactions on Power Delivery*, vol. 22, no. 3, pp. 1400–1410, 2007.

[6] A. M. Arasomwan and A. O. Adewumi, "An adaptive velocity particle swarm optimization for high-dimensional function optimization," in *Proceedings of the IEEE Congress Evolutionary Computation (CEC '13)*, pp. 2352–2359, 2013.

[7] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Anchorage, Alaska, USA, May 1998.

[8] G. I. Evers, *An automatic regrouping mechanism to deal with stagnation in particle swarm optimization [M.S. thesis]*, Graduate School of the University of Texas-Pan American, 2009.

[9] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming (EP '98)*, vol. 1447, pp. 591–600, San Diego, Calif, USA, March 1998.

[10] R. Akbari and K. Ziarati, "Combination of particle swarm optimization and stochastic local search for multimodal function optimization," in *Proceedings of the Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA '08)*, vol. 2, pp. 388–392, December 2008.

[11] C. Junying, Q. Zheng, L. Yu, and L. Jiang, "Particle swarm optimization with local search," in *Proceedings of the International Conference on Neural Networks and Brain (ICNNB '05)*, pp. 481–484, October 2005.

[12] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Improved particle swarm optimization combined with chaos," *Chaos, Solitons and Fractals*, vol. 25, no. 5, pp. 1261–1271, 2005.

[13] A. A. Mousa, M. A. El-Shorbagy, and W. F. Abd-El-Wahed, "Local search based hybrid particle swarm optimization algorithm for multiobjective optimization," *Swarm and Evolutionary Computation*, vol. 3, pp. 1–14, 2012.

[14] H. Pan, X. Han, and M. Zheng, "Particle swarm-simulated annealing fusion algorithm and its application in function optimization," in *Proceedings of the International Conference on Computer Science and Software Engineering (CSSE '08)*, vol. 21, pp. 78–81, December 2008.

[15] Y. Sun, B. J. Wyk, and Z. Wang, "A new golden ratio local search based particle swarm optimization," in *Proceedings of the International Conference on Systems and Informatics (ICSAI '12)*, pp. 754–757, 2012.

[16] J. Tang and X. Zhao, "Particle swarm optimization using adaptive local search," in *Proceedings of the International Conference on Future BioMedical Information Engineering (FBIE '09)*, pp. 300–303, December 2009.

[17] Y.-J. Wang, "Improving particle swarm optimization performance with local search for high-dimensional function optimization," *Optimization Methods and Software*, vol. 25, no. 5, pp. 781–795, 2010.

[18] W.-B. Zhang, J.-Y. Chen, and Y.-Q. Ye, "Study on particle swarm optimization algorithm with local interpolation search," in *Proceedings of the 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR '10)*, vol. 1, pp. 345–348, March 2010.

[19] A. M. Arasomwan and A. O. Adewumi, "On the performance of linear decreasing inertia weight particle swarm optimization for global optimization," *The Science World Journal*, vol. 2013, Article ID 860289, 2013.

[20] J. Ememipour, M. M. S. Nejad, M. M. Ebadzadeh, and J. Rezanejad, "Introduce a new inertia weight for particle swarm optimization," in *Proceedings of the IEEE 4th International Conference on Computer Sciences and Convergence Information Technology (ICCIT '09)*, pp. 1650–1653, November 2009.

[21] Y. Feng, G.-F. Teng, A.-X. Wang, and Y.-M. Yao, "Chaotic inertia weight in particle swarm optimization," in *Proceedings of the 2nd International Conference on Innovative Computing, Information and Control (ICICIC '07)*, September 2007.

[22] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing Journal*, vol. 11, no. 4, pp. 3658–3670, 2011.

[23] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the of the Congress on Evolutionary Computation*, vol. 1, pp. 94–100, Seoul, Korea, May 2001.

[24] M. E. H. Pedersen, *Tuning & simplifying heuristical optimization [Ph.D. thesis]*, School of Engineering Sciences, University of Southampton, Southampton, UK, 2010.

[25] S. Chetty and A. O. Adewumi, "Three new stochastic local search algorithms for continuous optimization problems," *Computational Optimization and Applications*, vol. 56, no. 3, pp. 675–721, 2013.

[26] D. Karaboga and B. Akay, "A comparative study of Artificial Bee Colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[27] B. A. Sawyerr, M. M. Ali, and A. O. Adewumi, "A comparative study of some real-coded genetic algorithms for unconstrained global optimization," *Optimization Methods and Software*, vol. 26, no. 6, pp. 945–970, 2011.

[28] C. Dytham, *Choosing and Using Statistics: A Biologist's Guide*, Wiley-blackwell, Malaysia, 3rd edition, 2011.

[29] J. Pallant, *SPSS Survival Manual*, McGraw-Hill, Singapore, 4th edition, 2010.

2013 IEEE Congress on Evolutionary Computation
June 20-23, Cancún, México

An Adaptive Velocity Particle Swarm Optimization for High-Dimensional Function Optimization

[1]Arasomwan Akugbe Martins and [2]Adewumi Aderemi Oluyinka
School of Mathematics, Statistics and Computer Science
University of Kwazulu-Natal
Durban, South Africa
[1]accuratesteps@yahoo.com, [2]adewumia@ukzn.ac.za

***Abstract*—Researchers have achieved varying levels of successes in proposing different methods to modify the particle's velocity updating formula for better performance of Particle Swarm Optimization (PSO). Variants of PSO that solved high-dimensional optimization problems up to 1,000 dimensions without losing superiority to its competitor(s) are rare. Meanwhile, high-dimensional real-world optimization problems are becoming realities hence PSO algorithm therefore needs some reworking to enhance it for better performance in handling such problems. This paper proposes a new PSO variant called Adaptive Velocity PSO (AV-PSO), which adaptively adjusts the velocity of particles based on Euclidean distance between the position of each particle and the position of the global best particle. To avoid getting trapped in local optimal, chaotic characteristics was introduced into the particle position updating formula. In all experiments, it is shown that AV-PSO is very efficient for solving low and high-dimensional global optimization problems. Empirical results show that AV-PSO outperformed AIWPSO, PSO_{rank}, CRIW-PSO, def-PSO, e1-PSO and APSO. It also performed better than LSRS in many of the tested high-dimensional problems. AV-PSO was also used to optimize some high-dimensional problems with 4,000 dimensions with very good results.**

Keywords—adaptive; global optimization; high dimension; optimization problems; particle swarm optimization; velocity updating

I. INTRODUCTION

It is almost two decades since PSO algorithm was introduced in [14] as a technique for solving optimization problems. It is one of the two fundamental mainstreams of swarm intelligence and it is driven by the simulation of a sociological metaphor inspired by collective behaviour of birds and other social organisms. Over the years, many researchers have made tremendous efforts to improve on the effectiveness, efficiency and robustness of PSO technique in solving optimization problems. Researches in this direction are the introduction of V_{max} [6], inertia weight [23], constriction factor [4] into PSO, as well as improvements on the inertia weight [1, 3, 7, 8, 11, 17, 18, 22, 24], PSO with mutation operators [2, 5, 11, 15, 16], hybridization of PSO with other algorithms [19] and development of other variants of PSO [21]. Despite these improvements on PSO, virtually none of the existing variants of PSO have been able to solve optimization problems with high dimension up to 2000. In [13] an improved PSO (IPSO) was used to solve Ackley's function with 100 dimensions and Sphere's function with 150 dimensions with evidences of superiority over the standard PSO. In [20] parallel PSO was used to optimize Griewank's function with 128 dimensions and there was evidence of success too. However in [12], a PSO algorithm was compared with Line Search Restart (LSRS) technique in solving some high-dimensional global optimization problems of dimension 50 to 2,000. In all the dimensions PSO lost its superiority to LSRS. Moreover, due to the fact that high dimensional real-world optimization problems is a possibility, PSO algorithm therefore need some reworking to enhance it for better performance in handling high-dimensional global optimization problems.

In this paper a very simple PSO algorithm (AV-PSO) is proposed. This work is different from existing ones in three major ways: firstly, it implemented the PSO algorithm without using any of inertia weight, acceleration coefficients and random coefficients to compute velocity for any particle in the swarm; secondly, chaotic characteristics was introduced into the particle's position formula to promote some stochasticity in order to facilitate good exploitation; thirdly, the proposed algorithm is used to favourably compete with another optimization algorithm (e.g. LSRS) to solve some optimization problems up to at least 2,000 dimensions. The rest of the paper is structured as follows: Section 2 succinctly looked at PSO. In section 3, the proposed simple PSO algorithm is described, while section 4 focuses on the numerical simulations. Section 5 briefly looked at the optimization characteristics of AV-PSO. Section 6 concludes this work summarizing the contributions made in this paper and future work.

II. PARTICLE SWARM OPTIMIZATION

A swarm of particles is involved in PSO. This swarm is randomly initialized as candidate solutions over the fitness landscape to start the process of optimizing a problem. Each particle is assumed to have position and velocity in a physical *d*-dimensional search space, the position and velocity of a particle i in each iteration t is represented as the vectors $\vec{X}_i = (x_{i1}, \ldots, x_{in})$ and $\vec{V}_i = (v_{i1}, \ldots, v_{in})$, respectively. When the particles move in the search space looking for the optimum solution for a particular optimization problem, other particles follow the current optimum particle by adjusting their *velocities* and *positions* using (1) and (2).

$$\vec{V}_i^{t+1} = \vec{V}_i^t + c_1\vec{r}_1\left(\overrightarrow{pbest}_i^t - \vec{X}_i^t\right) + c_2\vec{r}_2\left(\overrightarrow{gbest}_i^t - \vec{X}_i^t\right) \quad (1)$$

$$\vec{X}_i^{t+1} = \vec{X}_i^t + \vec{V}_i^{t+1} \quad (2)$$

where, $\vec{V}_i^t$ and $\vec{X}_i^t$ are the velocity and position of particle *i* at iteration *t*. A particle's position is taken as possible solution of the function being optimized while the fitness of this possible solution is determined by evaluating the function. The best position searched by the particle itself so far ($\overrightarrow{pbest}_i^t$) and the optimization position searched by the whole particles swarm so far ($\overrightarrow{gbest}_i^t$) are *d*-dimension vectors representing personal best position of particle *i* at iteration *t* and global best positions selected from the personal best positions of all the particles in the swarm at iteration *t*. Whereas, $\vec{r}_1$ and $\vec{r}_2$ are two *d*-dimensional vectors of random numbers between 0 and 1, which introduces randomness to the searching strategy and the two positive constants c_1 and c_2 are cognitive and social scaling parameters that determine the magnitude of the random forces in the direction of $\overrightarrow{pbest}_i^t$ and $\overrightarrow{gbest}_i^t$.

With the introduction of inertia weight by [23] into PSO, (1) was upgraded to become (3).

$$\vec{V}_i^{t+1} = \omega\vec{V}_i^t + c_1\vec{r}_1\left(\overrightarrow{pbest}_i^t - \vec{X}_i^t\right) + c_2\vec{r}_2\left(\overrightarrow{gbest}_i^t - \vec{X}_i^t\right) \quad (3)$$

Equation (3) is made up of inertia weight component $\omega\vec{V}_i^t$, cognitive component $c_1\vec{r}_1\left(\overrightarrow{pbest}_i^t - \vec{X}_i^t\right)$ and social component $c_2\vec{r}_2\left(\overrightarrow{gbest}_i^t - \vec{X}_i^t\right)$. The inertia weight strikes a balance between exploration and exploitation characteristics of PSO and it determines the level of contribution of previous particle velocity to the present velocity. It is a common idea among many researchers that, large inertia weight facilitates exploration while a small inertia weight facilitates exploitation; however there are cases where this may not hold [18].

The common practice has been to update particle velocity and position using (3) and (2) respectively. The goal of this research work is to propose very simple variants of PSO with fewer parameters to adjust, but with improved performance to achieve better convergence in few numbers of iterations compared with existing variants, without compromising quality results. In achieving this goal, the velocity for each particle was computed without using inertia weight (ω), acceleration coefficients (c_1 and c_2) and random parameters ($\vec{r}_1$ and $\vec{r}_2$) in equation (3). Rather, the velocity was adaptively adjusted based on Euclidean distance between the position of each particle and the position of the global best particle. To avoid getting trapped in local optimal, chaotic characteristics was introduced into (2). It is evident from experimental results that the proposed PSO variants, though simple in nature than many existing PSO variants, have tremendous performance in finding better global solutions to low and high-dimensional continuous optimization problems and achieves outstanding accuracy level of convergence in fewer numbers of iterations.

A. PSO Variants

Since the inception of PSO, quite a number of its variants have been proposed over the years. A comprehensive list of many of the variants can be found in [21]. The variants adopted for comparison in this work are Rank based PSO (PSO_{rank}) in [1], Adaptive Inertia Weight PSO (AIWPSO) in [18], Adaptive PSO (APSO) in [2], Natural Exponential inertia weight PSO (e_1-PSO) in [7], Decreasing exponential function PSO (def-PSO) in [8] and Chaotic Random Inertia Weight PSO (CRIW-PSO) in [10]. All these variants have proved to be superior to their competitors in their various capacities.

PSO_{rank} is based on cooperative behavior of particles. The local search and convergence to global optimum solution is controlled by selecting some number (which decreases with increased iteration) of best particles proportionate to their respective strengths, to contribute to the updating of the position of a candidate particle. The strength of each contributing particle is a function of strivness, immediacy and number of contributed particles. A time-varying inertia weight decreasing non-linearly was used to improve its performance. Experimental evidences show that PSO_{rank} is superior to its competitors [1].

AIWPSO uses the swarm success rate parameter to determine the inertia weight value. It monitors the search situation using the success rate to adapt the value of the inertia weight in the static and dynamic environment to improve its performance in dynamic environments. In this variant at the end of each iteration, the worst particle is replaced by mutated best particle. The mutation is done by adding a Gaussian noise with zero mean standard deviation σ to one of the randomly chosen dimension of the best particle to improve on exploration of the method. AIWPSO also outperformed its competitors virtually in all the tests [18]

The goal of APSO as proposed in [2] was to address the problem of unknown parameters estimation in nonlinear systems. It introduced an adaptive mutation mechanism and a dynamic inertia weight into the conventional PSO to enhance global search ability and to increase accuracy and was found to be more successful than the competing algorithms.

e_1-PSO is one of the two proposed natural exponential functions inertia weight strategies (which has some evidences of better performance than the other) in [3]. It is based on the basic idea of decreasing inertia weight. It was experimentally proved to converge faster in early stage of the search process and performed better in most continuous optimization problems than linear decreasing inertia weight PSO.

def-PSO adjusts the inertia weight based on decreasing exponential functions. The function is made up of two parts (base and power) and the algorithm's iteration was used in these parts. As the iteration increases, the inertia weight decreases from one towards zero. Graphical results in [8] show evidences of its superiority to the competitors.

In CRIW-PSO, two inertia weights (chaotic linear descending and chaotic random) based on the concept of linear descending and random inertia weights were proposed by introducing chaotic (using logistic mapping) optimization

mechanism into them. The purpose was to enhance the performance of PSO using linear descending and random inertia weights in terms of convergence precision, quick convergence velocity and better global search ability. Results in [10] show their outstanding performances.

Despite the various successes recorded by PSO variants, none of them have solved problems with up to 2000 dimensions. Besides, their results show that there are still needs for improvements on the convergence precision, quick convergence velocity, better global search ability, and further reduction of the possibilities of getting trapped in local optima by PSO algorithms. The goal of this paper is to experimentally address these issues by proposing another PSO variant.

III. PROPOSED PSO VARIANT

The velocity updating formula is a very important aspect of PSO algorithms. It determines the flying speed of particles in the search space when they are searching for optimal solutions to optimization problems. The proposed PSO algorithm in this paper updates the particle velocity and position using (4) and (7) respectively. It can quickly and efficiently locate better global optimal results for optimization problems without getting stuck in local optimal.

$$V_{i,t} = \frac{pDist_{i,t}}{maxDist_t} \tag{4}$$

where $pDist_{i,t}$ is the current Euclidean distance of particle i from the global best, at iteration t as shown in (5).

$$pDist_{i,t} = \sqrt{\sum_{j=1}^{d} \left(\overrightarrow{gbest_t^j} - \vec{X}_{i,t}^j\right)^2} \tag{5}$$

and $maxDist_t$ is the maximum distance, shown in (6), of a particle from the global best, at iteration t.

$$maxDist_t = \max_i \left(pDist_{i,t}\right) \tag{6}$$

$$\vec{X}_{i,t+1}^j = (1 - \tau)\vec{X}_{i,t}^j + V_{i,t} \tag{7}$$

where $\tau = 4 \times z \times (1 - z)$ is a logistic mapping and z is a random number in the interval (0,1). The chaotic mapping provides chaotic characteristics for the algorithm to escape premature convergence.

When any component of the position of any particle falls outside the search space, it is forced to take the value *maxX* (upper limit of search space) or *minX* (lower limit of search space) as the case may be.

The proposed PSO algorithm as is shown in Figure 1. The bold lines indicate the reworking done on the standard PSO algorithm.

Begin PSO Algorithm
Input: *f:* the function to optimize
ps: the swarm size
d: the problem dimension
Output: x^*: the best fitness value found
Initialize: $x_i = (x_{i1}, \ldots, x_{id})$ and $v_i = (v_{i1}, \ldots, v_{id})$, for all particles in problem space
evaluate $f(x_i)$ in d variables and get $pbest_i$, $(i = 1, \ldots, ps)$
$gbest \leftarrow$ best of $pbest_i$
While stopping criteria is false do
compute particles velocities using equation (4)
Begin Loop for *ps* times
do chaotic mapping
update x_i for particle using equation (7)
validate for position boundaries
End
If $f(x_i) < f(pbest_i)$ then
$pbest_i \leftarrow x_i$
end if
If $f(x_i) < f(gbest_i)$ then
$gbest_i \leftarrow x_i$
end if
End while
$x^* \leftarrow gbest$
Return x^*
End PSO Algorithm

Fig. 1. Pseudocode for the proposed PSO algoithm.

IV. NUMERICAL SIMULATION

Different test problems with varied difficulties were used to verify the performance of AV-PSO with its competitors. The simulations were carried out in four stages to test its convergence speed, accuracy, robustness, stability and global search ability in locating optimal values. In *stage 1*, AV-PSO was used to solve the test problems listed in [1, 2, 18] with same experimental settings, and results were compared with the existing results of PSOrank, AIWPSO and APSO recorded in literature. In *stage 2*, e1-PSO, def-PSO and CRIW-PSO were implemented and their performances compared with that of AV-PSO, using the same experimental settings and test problems. In *stage 3*, AV-PSO was compared with LSRS using several high-dimensional benchmark problems with dimensions from 50 to 2000 [12]. Finally in *stage 4*, AV-PSO was used to optimize the some problems with dimensions up to 4,000. A search goal of 0.0000000001 (10^{-10}) was used to test its efficiency relative to very high problems dimensionality.

The simulations were done on a laptop computer with a 2.0GHz Intel Pentium dual-core processor, 2.0GB of RAM, running Windows Vista Home Basic. The simulation program was developed using Microsoft Visual C# programming language, 2008 Express Edition.

A. Parameter settings

Refer to [1, 2, 18] for parameter settings for PSO_{rank}, AIWPSO and APSO and [12] for LSRS. In stage 2 of experiments, swarm size = 10, dimension = 2 for Schaffer's f6

and 40 for others, maximum iterations = 3000 and random runs = 50. In comparison with AIWPSO, AV-PSO used 60,000 function evaluations (FEs). The parameter settings for AV-PSO to test against LSRS are shown in Table I.

TABLE I. SETTINGS FOR AV-PSO IN COMPARISON WITH LSRS

Parameter	Number of dimensions				
	50	100	500	1000	2000
Swarm size	50	50	50	50	50
Maximum number of iterations	1000	1000	1000	1000	1000
Number of independent runs	25	25	25	25	25

B. Test problems

The test problems used in the experiments are made up of some standard continuous functions widely used in the literature. They were adopted from [1, 2, 12, 18] and any further details about them could be obtained from these references. Their characteristics are diverse enough to cover many of the problems which can arise in global optimization problems. AV-PSO was used to optimize these problems along with its competitors. All the problems were of dimension 40 with asymmetric initialization ranges. This dimension was chosen because test problems in stage 1 ended with dimension of 30 while stage 3 starts with 50.

1) Test problems of stage 1 of experiment

The test problems and their respective search as well as initialization ranges that were used to test whether AV-PSO could compete with existing PSO variants in literature were adopted from [1, 2, 18]. They have dimensions ranging from 2 – 30.

2) Test problems of stage 2 of experiment

To test the convergence speed, robustness as well as the global search ability of AV-PSO, the test problems given below were used. The performance of AV-PSO was compared with that of CRIW-PSO, def-PSO and e_1-PSO which were adopted for comparison.

Ackley (f_1): $f_1(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$

Search range: [-30,30], Initialization range: [15,30], Optimal value = 0

Griewank (f_2): $f_2(\vec{x}) = \frac{1}{4000}\left(\sum_{i=1}^{d} x_i^2\right) - \left(\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)\right) + 1$

Search range: [-600,600], Initialization range: [300,600], Optimal value = 0

Levy (f_3): $f_3(\vec{x}) = sin^2(\pi y_1) + \sum_{i=1}^{d-1}(y_i - 1)^2(1 + 10\,sin^2(\pi y_i + 1)) + (y_d - 1)^2(1 + sin^2(2\pi y_d)),$

where $y_i = 1 + \frac{x_i - 1}{4}$, $i = 1,2,\ldots,d$

Search range: [-10,10], Initialization range: [5,10], Optimal value = 0

Rastrigin (f_4): $f_4(\vec{x}) = \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$

Search range: [-5.12,5.12], Initialization range: [2.56,5.12], Optimal value = 0

Rosenbrock (f_5): $f_5(\vec{x}) = \sum_{i=1}^{d-1}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$

Search range: [-30,30], Initialization range: [15,30], Optimal value = 0

Schaffer's (f_6): $f_6(\vec{x}) = 0.5 + \frac{\left(sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2}$

Search range: [-100,100], Initialization range: [50,100], Optimal value = 0

3) Test problems of stage 3 of experiment

To test the capability of AV-PSO in solving high-dimensional problems, the test problems used by LSRS in [12] were also used. The performances of AV-PSO were compared with that of LSRS.

4) Test problems of stage 4 of experiment

In this stage, AV-PSO was used to optimize problems with the same search ranges in [12] with dimensions up to 4,000; but the Rastrigin of the test problems at stage 2 was used, while Quadric function was replaced with Griewank.

C. Experimental results and comparison

Shown in Tables II – XI are the numerical results for all the experiments that were performed in this work. The results for PSO_{rank}, AIWPSO and APSO were obtained from literature [1, 2, 18] while that of LSRS was adapted from [12] as recorded by the researchers, but the ones for CRIW-PSO, def-PSO, e_1-PSO and AV-PSO were obtained after the implementation of each of the algorithms.

1) Experimental results for stage 1

The experimental results are shown in Tables II – IV with Test problems (TP), Population size (PS), Problem dimension (PD) and Iteration (ITR). Best optimal results appear in bold.

Table II reflects the performance measurement of PSO_{rank} and AV-PSO in six test problems with three different problem dimensions over 100 independent runs. From results in the table, the two algorithms had equal performances in f_3 across the three dimensions in both average fitness and standard deviation as well as in f_6. In all other test problems across the dimensions, AV-PSO extremely outperformed PSO_{rank} in convergence speed, accuracy, robustness and stability with better global search ability and location of optimal values. Except f_2, AV-PSO obtained the minimum value for all the test problems. Also the performance of PSO_{rank} decreases with increasing problem dimension in f_1, f_2, and f_4, but AV-PSO do not which is an indication that it is less sensitive to increase in problem dimensionality compared with PSO_{rank}. The two algorithms had equal success rates in other test problems except in f_2 where AV-PSO outperformed PSO_{rank} in all dimensions. After the maximum number of iterations, an algorithm was considered successful if the minimum value reached was below 0.000001 (10^{-6}) for f_6, and 0.01 (10^{-2}) for others.

From the results in Table III, it is clear that AV-PSO is superior in all the test problems to APSO in performance. The performance measurements are in term of average fitness values of the best particle and standard deviation. The standard deviation was computed for dimension 10 across the

population sizes for the three test problems [2]. The results show that AV-PSO demonstrates superiority over APSO in global searching abilities, stability and robustness. Also, the convergence velocity and precision of AV-PSO within the given iterations are far higher than that of APSO. The results reveal that APSO is sensitive to problems dimensions and swarm sizes in all the test problems because as the number of particles increases the quality of fitness also increase across the dimensions, but the case of AV-PSO is not so as the results obtained by it remain stable and the minimum optimal value for all the test problems which is a clear indication of its stability and robustness.

TABLE II. PERFORMANCE COMPARISON BETWEEN PSO_{RANK} AND AV-PSO

PD	TP	Average fitness		Standard deviation		Success rate	
		PSO_{rank}	AV-PSO	PSO_{rank}	AV-PSO	PSO_{rank}	AV-PSO
10	f_1	1.21E-10	**0.00E+00**	8.36E-10	**0.00E+00**	1	1
	f_2	9.14E-03	**2.37E-11**	1.42E-02	**8.78E-11**	0.96	1
	f_3	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	1	1
	f_4	1.31E-06	**0.00E+00**	6.54E-06	**0.00E+00**	1	1
	f_5	2.53E-05	**0.00E+00**	3.47E-05	**0.00E+00**	1	1
	f_6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1	1
20	f_1	1.08E-09	**0.00E+00**	3.76E-09	**0.00E+00**	1	1
	f_2	1.61E+00	**1.75E-11**	2.04E+00	**1.35E-10**	0.56	1
	f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1	1
	f_4	4.22E-06	**0.00E+00**	9.11E-06	**0.00E+00**	1	1
	f_5	4.47E-07	**0.00E+00**	7.69E-07	**0.00E+00**	1	1
30	f_1	2.05E-08	**0.00E+00**	6.41E-08	**0.00E+00**	1	1
	f_2	1.27E+01	**1.83E-12**	1.39E+01	**5.01E-12**	0.19	1
	f_3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1	1
	f_4	3.12E-05	**0.00E+00**	8.35E-05	**0.00E+00**	1	1
	f_5	2.73E-08	**0.00E+00**	5.24E-08	**0.00E+00**	1	1

Note: Sphere (f_1), Rosenbrock (f_2), Rastrigin (f_3), Ackley (f_4), Griewank (f_5), Schaffer's f6 (f_6).

TABLE III. PERFORMANCE COMPARISON BETWEEN APSO AND AV-PSO

TP	PS	PD	ITR	Average fitness		Standard deviation	
				APSO	AV-PSO	APSO	AV-PSO
f_1	20	10	1 000	0.0983	**0.0000**	0.0054	**0.0000**
		20	1 500	0.0237	**0.0000**		
		30	2 000	0.0117	**0.0000**		
	40	10	1 000	0.0952	**0.0000**	0.0036	**0.0000**
		20	1 500	0.0201	**0.0000**		
		30	2 000	0.0105	**0.0000**		
	80	10	1 000	0.0689	**0.0000**	0.0029	**0.0000**
		20	1 500	0.0199	**0.0000**		
		30	2 000	0.0102	**0.0000**		
f_2	20	10	1 000	5.1565	**0.0000**	0.1358	**0.0000**
		20	1 500	16.0456	**0.0000**		
		30	2 000	42.2325	**0.0000**		
	40	10	1 000	2.9468	**0.0000**	0.1064	**0.0000**
		20	1 500	15.3678	**0.0000**		
		30	2 000	33.7538	**0.0000**		
	80	10	1 000	2.0457	**0.0000**	0.1084	**0.0000**
		20	1 500	10.0563	**0.0000**		
		30	2 000	25.3473	**0.0000**		
f_3	20	10	1 000	5.8467	**0.0000**	1.3471	**0.0000**
		20	1 500	47.9842	**0.0000**		
		30	2 000	100.4528	**0.0000**		
	40	10	1 000	4.5431	**0.0000**	1.2376	**0.0000**
		20	1 500	38.3464	**0.0000**		
		30	2 000	72.5473	**0.0000**		
	80	10	1 000	4.1680	**0.0000**	1.1450	**0.0000**
		20	1 500	27.9547	**0.0000**		
		30	2 000	69.0609	**0.0000**		

Note: Griewank (f_1), Rastrigin (f_2), Rosenbrock (f_3)

Shown in Table IV is the performance measurement of AIWPSO and AV-PSO in thirteen test problems over 30 independent runs. Better optimal values appear in bold. From results, the two algorithms had equal performances in f_7 in both mean fitness and standard deviation. In all other test problems, AV-PSO extremely outperformed AIWPSO in convergence speed, accuracy, robustness and stability with better global search ability and location of optimal values except f_8 and f_{13}. Out of the thirteen problems, AV-PSO obtained outright minimum values for eight while AIWPSO obtained for only one.

TABLE IV. PERFORMANCE COMPARISON BETWEEN AIWPSO AND AV-PSO

TP	Mean fitness		Standard deviation	
	AIWPSO	AV-PSO	AIWPSO	AV-PSO
f_1	3.3703E-134	**0.0000E+000**	5.1722E-267	**0.0000E+000**
f_2	1.8317E-137	**0.0000E+000**	3.4534E-273	**0.0000E+000**
f_3	1.6534E-062	**0.0000E+000**	7.7348E-123	**0.0000E+000**
f_4	1.9570E-010	**0.0000E+000**	1.2012E-019	**0.0000E+000**
f_5	5.5241E-003	**1.2186E-003**	**1.5358E-005**	5.4511E-003
f_6	2.5003E+000	**6.9252E-013**	1.5978E+001	**1.7255E-012**
f_7	0.0000E+000	0.0000E+000	0.0000E+000	**0.0000E+000**
f_8	**-1.1732E+004**	-1.9669E+003	**1.1409E-025**	5.0474E+002
f_9	1.6583E-001	**0.0000E+000**	2.1051E-001	**0.0000E+000**
f_{10}	1.1842E-016	**0.0000E+000**	4.2073E-031	**0.0000E+000**
f_{11}	6.9870E-015	**4.4409E-016**	4.2073E-031	**0.0000E+000**
f_{12}	2.8524E-002	**0.0000E+000**	7.6640E-004	**0.0000E+000**
f_{13}	**1.4998E-032**	1.1570E-014	**1.2398E-094**	3.1485E-014

Note: Sphere (f_1), Sphere (f_2), Schwefel P2.22 (f_3), Rotated hyper-ellipsoid (f_4), Noisy Quadric (f_5), Rosenbrock (f_6), Step (f_7), Schwefel (f_8), Rastrigin (f_9), Non continuous Rastrigin (f_{10}), Ackley (f_{11}), Griewank (f_{12}), Levy (f_{13}).

2) Expeimental results for stage 2

In this section, the performance of AV-PSO is further tested to prove its superiority in convergence speed, accuracy, robustness, stability, and global search ability in locating optimal values. Table V with Test problems (TP), Population size (PS), Problem dimension (PD), Iteration (ITR) shows the results of all the algorithms after implementing them. The comparisons criterions are mean fitness value, Standard deviation (SD) and Success rate. After the maximum number of iterations, an algorithm was considered successful if the minimum value reached was below 0.00000001 (10^{-8}) for all the test problems.

The results in Table V indicate that AV-PSO is extremely superior to its competitors in all the test problems in every way. Again, it is superior in convergence speed, accuracy, robustness, stability and global search ability in locating optimal values. In $f_1 - f_5$, it is only AV-PSO that met the success criterion in all the runs. Though, in f_6 two of the competitors were able to meet the success criterion but with lower SR compared with AV-PSO.

3) Experimental results for stage 3

After subjecting AV-PSO to various swarm sizes, problem dimensionality, iterations and independent runs to prove its performance in the preceding tests, it is finally compared with LSRS using various high-dimensional problems with varying complexities to test its capability in solving such problems. The results are categorized in problem dimensions in Tables VI – X.

TABLE V. RESULTS OF AV-PSO AND THE COMPETING VARIANTS OVER 50 INDEPENDENT RUNS. "X" INDICATES NO RESULT

TP	Comparison index		CRIW-PSO	def-PSO	e_1-PSO	AV-PSO
f_1	Fitness value	Best	1.92E+01	1.97E+01	1.95E+01	4.44E-16
		Worst	1.99E+01	2.03E+01	1.99E+01	4.44E-16
		Mean	1.98E+01	1.99E+01	1.98E+01	**4.44E-16**
		SD	1.15E-01	1.18E-01	8.20E-02	**0.00E+00**
	Iterations to achieve set goal	Best	x	x	x	6
		Worst	x	x	x	18
		Mean	x	x	x	11
	Success Rate		0	0	0	**100%**
f_2	Fitness value	Best	1.80E+02	3.30E+02	2.70E+02	0.00E+00
		Worst	9.91E+02	9.37E+02	1.08E+03	0.00E+00
		Mean	5.79E+02	6.07E+02	6.53E+02	**0.00E+00**
		SD	1.94E+02	1.38E+02	1.73E+02	**0.00E+00**
	Iterations to achieve set goal	Best	x	x	x	3
		Worst	x	x	x	10
		Mean	x	x	x	6
	Success Rate		0	0	0	**100%**
f_3	Fitness value	Best	3.68E+01	8.35E+01	1.77E+01	4.99E-22
		Worst	2.16E+02	2.40E+02	2.25E+02	5.87E-13
		Mean	1.02E+02	1.47E+02	1.06E+02	**2.18E-14**
		SD	4.39E+01	4.48E+01	4.87E+01	**8.81E-14**
	Iterations to achieve set goal	Best	x	x	x	4
		Worst	x	x	x	205
		Mean	x	x	x	56
	Success Rate		0	0	0	**100%**
f_4	Fitness value	Best	2.93E+02	3.63E+02	2.90E+02	0.00E+00
		Worst	5.11E+02	6.52E+02	5.23E+02	0.00E+00
		Mean	3.81E+02	4.93E+02	4.09E+02	**0.00E+00**
		SD	5.46E+01	6.12E+01	5.12E+01	**0.00E+00**
	Iterations to achieve set goal	Best	x	x	x	3
		Worst	x	x	x	19
		Mean	x	x	x	9
	Success Rate		0	0	0	**100%**
f_5	Fitness value	Best	3.95E+01	3.88E+07	6.40E+02	3.31E-18
		Worst	4.00E+08	3.61E+08	4.80E+08	7.72E-11
		Mean	1.83E+08	1.67E+08	2.67E+08	**3.00E-12**
		SD	1.14E+08	7.23E+07	1.05E+08	**1.14E-11**
	Iterations to achieve set goal	Best	x	x	x	1
		Worst	x	x	x	1,052
		Mean	x	x	x	216
	Success Rate		0	0	0	**100%**
f_6	Fitness value	Best	0.00E+00	2.28E-01	0.00E+00	0.00E+00
		Worst	9.72E-03	5.00E-01	4.98E-01	1.49E-02
		Mean	4.66E-03	4.52E-01	3.55E-01	**2.98E-04**
		SD	4.85E-03	8.40E-02	1.70E-01	**2.08E-03**
	Iterations to achieve set goal	Best	122	x	221	3
		Worst	2,184	x	2,504	2,091
		Mean	843	x	842	54
	Success Rate		52%	0	12%	**98%**

From the results in Tables VI – X, it can be seen that LSRS and the proposed PSO algorithm (AV-PSO) put up a good competition solving some high-dimensional optimization problems. In all the dimensions, AV-PSO performed better than LSRS in 4 out of the 7 problems because it was able to obtain the minimum optimal values for Quadric, Schwefel, Sphere and Sum squares problems. However, in Ackley, Levy and Rosenbrock problems, the results obtained by LSRS were higher in magnitude compared with AV-PSO which also obtained good results. AV-PSO obtained its results with a small swarm size and fewer numbers of iterations compared with LSRS. It is very possible that AV-PSO could outsmart LSRS solving Ackley, Levy and Rosenbrock problems if AV-PSO uses some larger population size than 50 and maximum iterations than 1,000 used in this experiment. These results obtained at this stage 3 are indications that PSO algorithm has the potentials to solve many high-dimensional optimization problems without losing its superiority to its competitor(s).

4) Experimental esults for stage 4

In Table XI are the results of AV-PSO when it was used to optimize high-dimensional optimization problems with 4,000 dimensions over 25 independent runs, to show that it can handle more than 2,000 dimensions. This also help to find out the effect of further increase in problem dimensions would have on its performance. Best, worst and average numbers of iterations and Function Evaluations were rounded up to the nearest integer values.

From the results shown in Tables XI, it is evident that the proposed algorithm (AV-PSO) can efficiently optimize high-dimension optimization problems up to 4,000 dimensions. The best, worst and average number of iterations the algorithm obtained optimal values less than 10^{-10} shows that AV-PSO has a very high convergence speed and accuracy. It was able to the minimum optimal results for 5 out of the 8 problems, even with such a high criterion of 10^{-10}. The algorithm is still very strong enough to handle dimensions greater than 4000 with very good results. The algorithm is robust, efficient and has very good global search ability despite the fact that the solution spaces of the problems increased exponentially with their problem sizes

TABLE VI. THE PERFORMANCE OF LSRS AND AV-PSO FOR 50 DIMENSIONS

Algorithm	Comparison index	Test Problems						
		Ackley	Levy	Quadric	Rosenbrock	Schwefel	Sphere	Sum squares
LSRS	Best	−6.50E−19	2.90E−39	6.27E−19	2.47E−28	1.86E−11	1.34E−22	9.86E−21
	Average	−6.50E−19	2.90E−39	2.33E−18	1.38E−18	1.91E−11	1.38E−18	1.42E−18
	SD	0.00E+00	0.00E+00	8.11E−19	1.29E−18	4.15E−12	1.29E−18	1.25E−18
AV-PSO	Best	4.44E−16	4.75E−20	0.00E+00	1.55E−17	0.00E+00	0.00E+00	0.00E+00
	Average	5.86E−16	4.87E−13	0.00E+00	6.70E−11	0.00E+00	0.00E+00	0.00E+00
	SD	6.96E−16	1.32E−12	0.00E+00	2.51E−10	0.00E+00	0.00E+00	0.00E+00

TABLE VII. THE PERFORMANCE OF LSRS AND AV-PSO FOR 100 DIMENSIONS

Algorithm	Comparison index	Test Problems						
		Ackley	Levy	Quadric	Rosenbrock	Schwefel	Sphere	Sum squares
LSRS	Best	−6.50E−19	2.90E−39	9.20E−16	5.83E−28	7.81E−19	5.34E−19	4.68E−18
	Average	−6.50E−19	2.90E−39	1.15E−15	6.94E−16	3.98E−10	6.94E−16	6.98E−16
	SD	0.00E+00	0.00E+00	4.38E−16	6.63E−16	4.97E−10	6.63E−16	6.58E−16
AV-PSO	Best	4.44E−16	3.60E−28	0.00E+00	1.60E−25	0.00E+00	0.00E+00	0.00E+00
	Average	4.44E−16	3.03E−13	0.00E+00	6.33E−10	0.00E+00	0.00E+00	0.00E+00
	SD	0.00E+00	7.37E−13	0.00E+00	2.71E−10	0.00E+00	0.00E+00	0.00E+00

TABLE VIII. THE PERFORMANCE OF LSRS AND AV-PSO FOR 500 DIMENSIONS

Algorithm	Comparison index	Test Problems						
		Ackley	Levy	Quadric	Rosenbrock	Schwefel	Sphere	Sum squares
LSRS	Best	−4.30E−19	2.90E−39	2.12E−11	3.40E−27	2.91E−19	4.54E−16	4.05E−35
	Average	−4.30E−19	2.90E−39	4.31E−11	2.61E−11	4.08E−19	9.00E−16	7.96E−35
	SD	0.00E+00	0.00E+00	1.14E−11	2.32E−11	3.62E−20	1.52E−16	1.95E−35
AV-PSO	Best	4.44E−16	1.53E−21	0.00E+00	1.66E−17	0.00E+00	0.00E+00	0.00E+00
	Average	5.06E−16	5.28E−13	0.00E+00	7.45E−11	0.00E+00	0.00E+00	0.00E+00
	SD	6.96E−16	1.71E−12	0.00E+00	1.67E−10	0.00E+00	0.00E+00	0.00E+00

TABLE IX. THE PERFORMANCE OF LSRS AND AV-PSO FOR 1000 DIMENSIONS

Algorithm	Comparison index	Test Problems						
		Ackley	Levy	Quadric	Rosenbrock	Schwefel	Sphere	Sum squares
LSRS	Best	1.30E−18	2.90E−39	5.34E−30	6.84E−27	9.30E−18	7.97E−19	3.78E−33
	Average	1.30E−18	2.90E−39	1.38E−29	7.41E−27	1.12E−17	1.25E−18	7.35E−33
	SD	4.80E−33	0.00E+00	3.68E−30	1.66E−28	7.33E−19	2.05E−19	1.49E−33
AV-PSO	Best	4.44E−16	2.58E−18	0.00E+00	7.63E−18	0.00E+00	0.00E+00	0.00E+00
	Average	5.86E−16	4.40E−12	0.00E+00	1.47E−10	0.00E+00	0.00E+00	0.00E+00
	SD	6.96E−16	1.43E−11	0.00E+00	3.23E−10	0.00E+00	0.00E+00	0.00E+00

TABLE X. THE PERFORMANCE OF LSRS AND AV-PSO FOR 2000 DIMENSIONS

Algorithm	Comparison index	Test Problems						
		Ackley	Levy	Quadric	Rosenbrock	Schwefel	Sphere	Sum squares
LSRS	Best	−4.30E−19	2.9E−39	9.37E−08	1.40E−26	2.41E−17	9.97E−34	7.58E−31
	Average	−4.30E−19	2.9E−39	1.69E−07	1.48E−26	3.08E−17	2.35E−33	1.27E−30
	SD	9.60E−35	0.00E+00	3.42E−08	2.44E−28	2.31E−18	6.91E−34	2.02E−31
AV-PSO	Best	4.44E−16	8.11E−19	0.00E+00	2.49E−15	0.00E+00	0.00E+00	0.00E+00
	Average	7.28E−16	2.54E−11	0.00E+00	1.68E−09	0.00E+00	0.00E+00	0.00E+00
	SD	9.64E−16	7.31E−11	0.00E+00	5.83E−09	0.00E+00	0.00E+00	0.00E+00

TABLE XI. THE PERFORMANCE OF AV-PSO FOR 4000 DIMENSIONS OVER 25 RUNS WITH SIZE = 50, MAXIMUM ITERATION = 1,000

Algorithm	Measurement Index		Test Problems							
			Ackley	Griewank	Levy	Rastrigin	Rosenbrock	Schwefel	Sphere	Sum squares
AV-PSO	Fitness Values	Best	4.44E−16	0.00E+00	3.17E−19	0.00E+00	3.25E−14	0.00E+00	0.00E+00	0.00E+00
		Worst	4.00E−15	0.00E+00	3.00E−10	0.00E+00	2.80E−08	0.00E+00	0.00E+00	0.00E+00
		Average	7.28E−16	0.00E+00	2.62E−11	0.00E+00	1.41E−09	0.00E+00	0.00E+00	0.00E+00
		SD	9.64E−16	0.00E+00	6.80E−11	0.00E+00	5.47E−09	0.00E+00	0.00E+00	0.00E+00
	Iterations to obtain fitness values (less than 10^{-10})	Best	8	3	18	4	56	11	3	6
		Worst	18	14	940	16	977	23	14	17
		Average	12	7	213	9	582	16	8	11
	Success Rate		100%	100%	92%	100%	68%	100%	100%	100%
	Function Evaluations		620	328	11,550	470	42,772	798	424	534

V. THE OPTIMIZATION CHARACTERISTICS OF AV-PSO

Fig. 2 and 3, show the adaptive nature of the rate at which each particle moves towards the global best and chaotic weighting values (1 – Z) of each particle's position when optimizing Rastrigin function with 30 dimensions and 30 particles. In iteration 1, particle 4 is the global best and because others are far away from it they need high velocity relative to their positions to move towards particle 4. Their velocity reduces relative to their positions as they all converge towards the optimal point while exploiting their neighbourhoods for better results during iterations 40 and 200. The possibilities of moving too fast and jumping over the global best or moving too slow and not reaching the global best is not experienced because of the adaptivity. The chaotic weighting helps the particles for better exploration/exploitation and escape from local optimal. These are the reasons behind the outstanding successes of AV-PSO.

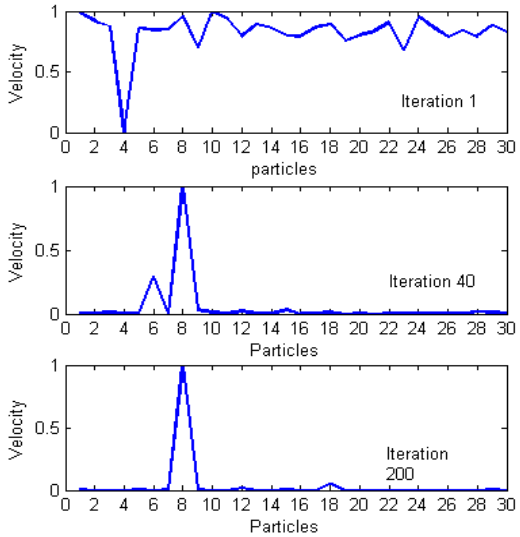


Fig. 2. Velocity of each particle during iterations

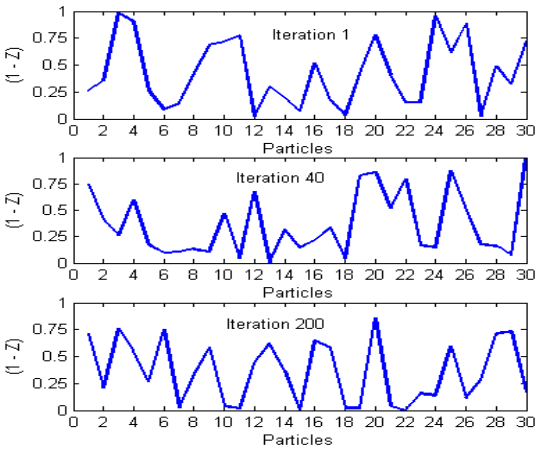


Fig. 3. Chaotic weighting of each particle during iterations

VI. Conclusion and Future Work

A very simple but effective PSO variant (AV-PSO) was proposed in this paper. The inertia weight, acceleration coefficients and random factors were found not to be necessary in the particle velocity updating equation for the algorithm to obtain outstanding and accurate global optimal solutions for low and high-dimensional test problems. In all the experiments AV-PSO extremely outperformed all its competitors, solving continuous optimization problems with low (10 – 30) and high (50 – 4,000) dimensions. With the algorithm, this work has experimentally shown that PSO is very much suitable for large-scale global optimization problems involving very high dimensions, with very good performance in locating quality global optimal solutions with few numbers of iterations without getting stuck in local optimal.

The algorithms have the potentiality to solve problems with higher number of variables greater 4,000 without any modifications. Further work shall be done by applying the proposed algorithm to more difficult continuous problems and discrete problems. Also, various information through numerical experiments shall be retrieved for analysis to find further reasons for the outstanding performances of the proposed PSO variant.

Acknowledgement

Our thanks to the College of Agricultural Science, Engineering and Sciences, University of Kwazulu-Natal, South Africa for their support towards this work through financial bursary.

References

[1] R. Akbari, and K. Ziarati, "A rank based particle swarm optimization algorithm with dynamic adaptation". Journal of Computational and Applied Mathematics, Elsevier, vol. 235, pp. 2694–2714, 2011.

[2] A. Alfi, "PSO with adaptive mutation and inertia weight and its application in parameter estimation of dynamic systems". ACTA, Automatic Sinica, vol. 37, no. 5, pp. 541-549, May 2011.

[3] G. Chen, X. Huang, J. Jia, and Z. Min., "Natural exponential inertia weight strategy in particle swarm optimization", The Sixth World Congress on Intelligent Control and Automation, IEEE, vol. 1, pp. 3672–3675, 2006.

[4] M. Clerc, and J. Kennedy, "The particle swarm - explosion, stability, and convergence in multidimensional complex space". IEEE Transactions on Evolutionary Computation,vol. 6, pp. 58-73, Feb. 2002.

[5] L.D.S. Coelho, "A quantum particle swarm optimizer with chaotic mutation operator", Chaos, Solutions and Fractals, Elsevier, vol. 37, pp. 1409-1418, 2008.

[6] R.C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory". Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95, pp. 39-43, Nagoya, Japan, 1995.

[7] R.C. Eberhart, and Y. Shi., "Tracking and optimizing dynamic systems with particle swarms," Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 94–100, 2002.

[8] J. Ememipour, M.M.S. Nejad, M.M. Ebadzadeh, and J. Rezanejad, "Introduce a new inertia weight for particle swarm optimization", The Fourth International Conference on Computer Sciences and Convergence Information Technology, pp. 1650-1653. IEEE, 2009.

[9] G.I. Evers, "An automatic regrouping mechanism to deal with stagnation in particle swarm optimization," MSc. Thesis, Graduate school of the University of Texas-Pan American, May, 2009.

[10] Y. Feng, G.F. Teng, A.X. Wang, and Y.M. Yao., "Chaotic Inertia Weight in Particle Swarm Optimization," Second International Conference on Innovative Computing, Information and Control, IEEE, p. 475, 2008.

[11] Y. Gao, X. An, and J. Liu., "A particle swarm optimization algorithm with logarithm decreasing inertia weight and chaos mutation", International Conference on Computational Intelligence and Security, IEEE, vol. 1, pp. 61–65, 2008.

[12] C. Grosan, and A. Abraham, "A novel global optimization technique for high dimensional functions". International journal of intelligent systems, Wiley Periodicals, Inc., vol. 24, pp. 421-440, 2009.

[13] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm", Chaos, solutions and fractals, Elsevier, no. 37, pp. 698-705, 2006.

[14] J. Kennedy, and R.C. Eberhart, "Particle swarm optimization", Proceedings of IEEE international conference on neural networks, Perth, Australia,vol. 4, pp. 1942–1948, 1995.

[15] H.R. Li and Y.L. Gao., "Particle swarm optimization algorithm with exponent decreasing inertia weight and stochastic mutation", Second International Conference on Information and Computing Science, IEEE, pp. 66–69, 2009.

[16] Li-Lili, and He-Xingshi, "Gaussian mutation Particle Swarm Optimization with dynamic adaptation inertia weight", World Congress on Software Engineering, IEEE, pp. 454-459, 2009.

[17] R.F. Malik, T.A. Rahman, S.Z.M. Hashim, and R. Ngah, "New particle swarm optimizer with sigmoid increasing inertia weight," International Journal of Computer Science and Security, vol. 1, no. 2, pp. 35-44, 2007.

[18] A. Nickabadi , M.M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," Applied soft computing on, vol. 11, pp. 3658-3670, 2011.

[19] V.M. Saffarzadeh, P. Jafarzadeh, and M. Mazloom, "A hybrid approach using particle swarm optimization and simulated annealing for n-queen problem", World academy of science, engineering and technology, vol. 67, pp. 517-521, 2010.

[20] J. F. Schutte, J.A. Reinbolt, B.J. Fregly, R.T. Haftka, and A.D.George, "Parallel global optimization with the particle swarm algorithm", International journal for numerical methods in engineering, John Wiley & Sons, Ltd., vol. 61, pp. 2296–2315, Oct. 2004.

[21] D. Sedighizadeh, and E. Masehian, "Particle swarm optimization methods, taxonomy and applications", International journal of computer theory and engineering, vol. 1, no. 5, pp. 486-502, Dec. 2009.

[22] X. Shen, Z. Chi, J. Yang, C. Chen, and Z. Chi, "Particle swarm optimization with dynamic adaptive inertia weight", International Conference on Challenges in Environmental Science and Computer Engineering, pp. 287-290. IEEE, 2010.

[23] Y. Shi, and R. C. Eberhart, "A modified particle swarm optimizer," Proceedings of the IEEE international conference on evolutionary computation, pp. 69–73, 1998.

[24] J. Xin, G. Chen, and Y. Hai , "A particle swarm optimizer with multi-stage linearly-decreasing inertia weight," International Joint Conference on Computational Sciences and Optimization, pp. 505-508, 2009.