THE FUSION OF PARTICLE SWARM OPTIMIZATION (PSO) AND INTERIOR POINT METHOD (IPM) AS COOPERATIVE MOVEMENT CONTROL ALGORITHM IN SWARM ROBOTICS

DADA EMMANUEL GBENGA

FACULTY OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

UNIVERSITY OF MALAYA

2016

THE FUSION OF PARTICLE SWARM OPTIMIZATION (PSO) AND INTERIOR POINT METHOD (IPM) AS COOPERATIVE MOVEMENT CONTROL ALGORITHM IN SWARM ROBOTICS

DADA EMMANUEL GBENGA

THESIS SUBMITTED AS FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND

INFORMATION TECHNOLOGY

UNIVERSITY OF MALAYA

2016

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **DADA EMMANUEL GBENGA**

Registration /Matric No: **WHA120022**

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Project Paper/Research Report/Dissertation/Thesis ("this work"): **The Fusion of Particle Swarm Optimization (PSO) and Interior Point Method Algorithms for Swarm Robotics**

Field of Study: **Swarm Intelligence (Computer Science)**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this work;

(2) This work is original;

(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any except or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this work;

(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;

(5) I hereby assign all and every rights in the copyright to this work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;

(6) I am fully aware that if in the course of making this work I have infringed any copyright whether intentionally or otherwise, I may be subjected to legal action or any other action as may be determined by UM.

Candidate's Signature Date

Subscribed and solemnly declared before,

Witness's Signature Date

Name:

Designation:

ABSTRACT

Research in Particle Swarm Optimisation and its applications to real world problems has become a very interesting field in recent years. Particle Swarm Optimisation (PSO) despite its simplicity, ease of implementation and efficiency still has some flaws, which include its tendency to premature convergence and inability to escape local minima. To address these weaknesses, many variants of PSO have been proposed in the literature. Also, many of these PSO algorithms employed hybrid methods that integrate other optimisation algorithms with the standard PSO. It is demonstrated in the literature that methods that hybridize PSO and some other optimisation algorithm have a better performance over the standard PSO algorithm. The Primal Dual method have been used to solve many optimisation problems.

We proposed the Primal-Dual Particle Swarm Optimisation (*pdPSO*) and Primal-Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*) to resolve the shortcomings of the standard PSO without the limitations of the IPM methods. To evaluate the performance of our new algorithms, we first compared the performance of *pdPSO* with IPM and PSO using nine (9) different dynamic benchmark functions. Our results revealed that *pdPSO* performed better than both the conventional PSO algorithm and the IPM method. The proposed algorithm is not susceptible to premature convergence, and can handle local minima avoidance better when compared to conventional PSO. Hence, *pdPSO* has the potential to perform better than many other PSO variants. Secondly, we compared the performance of our new algorithm *pdAPSO* with APSO, and PSO using 7 benchmark functions. Optimisation results reveal that *pdAPSO* offers similar (or in many test cases better) solutions than the other PSO variants to which we compared. Thirdly, we make a comparison between the performance of *pdPSO* and *pdAPSO*. Finally, we used our hybrid algorithms (*pdPSO* and *pdAPSO*) to solve the flocking and pattern formation problem in swarm robotics. Our simulation result

provides a clear indication of the effectiveness of the algorithm. The hybrid algorithms perform better in terms of precision, rate of convergence, steadiness, robustness and flocking capability for homogenous set of swarm robots compared to some other variants of PSO.

We also compared the performance of *pdAPSO* and *pdPSO* with 9 state of the art PSO algorithms using 12 benchmark functions. Our proposed algorithms have mean dependability of 80.4% for *pdAPSO* and 69.69% for *pdPSO*. Also, *pdAPSO* and *pdPSO* is a better convergence speed compared to the other 9 algorithms. For instance, on Rosenbrock function, the mean FEs of 8938, 6786, 10,080, 9607, 11,680, 9287, 23,940, 6269 and 6198 are required by PSO-LDIW, CLPSO, pPSA, PSOrank, OLPSO-G, ELPSO, APSO-VI, DNSPSO and MSLPSO respectively to get to the global optima. However, *pdPSO* and *pdAPSO* only use 2997 and 2124 respectively which shows that *pdAPSO* is the fastest convergence speed and closely followed by *pdPSO*. In summary, *pdPSO* and *pdAPSO* uses the lowest number of FEs to arrive at acceptable solutions for all the 12 benchmark functions.

ABSTRAK

Penyelidikan dalam "Particle Swarm Optimisation" dan aplikasinya kepada masalah dunia sebenar telah menjadi satu bidang yang sangat menarik sejar kebelakangan ini. Antara teknik-teknik pengoptimuman yang wujuo, "Particle Swarm Optimisation" (PSO) adalah salah satu yang paling popular kerana kesederhanaan nya mudah dilaksana kan dan kecekapan. Algoritma tersebut Bagaimanapun, algorithma ini mempunyai beberapa kelemahan, seperti kecenderungan untuk penumpuan pra-matang dan ketidakupayaan untuk melarikan diri dari terperangkap bi dalam minima tempatan

Ia juga telah digunakan untuk menyelesaikan fungsi kos yang berbeza tak linear dan bukan licin yang tersebar luas dalam reka bentuk rangkaian, pembinaan semula imej perubatan dan kejuruteraan industri. Ianya mempunyai keupayaan untuk menangani masalah pelbagai dimensi pengan berkesan. Kami berhasrat untuk menggabungkan kedua-dua algoritma ini untuk menghasilkan satu set algoritma PSO hibrid yang akan dapat menyelesaikan masalah-masalah yang dinyatakan di atas yang berkaitan dengan PSO. Kami mencadangkan primitif-Dual "Particle Swarm Optimisation" (*pdPSO*) dan "Primal-Dual Asynchronous Particle Swarm Optimisation" (*pdAPSO*) untuk menyelesaikan kelemahan PSO asli dan juga tanpa batasan kaedah IPM tanpa batasan. Integrasi ini melahirkan sistem yang mempunyai kapasiti yang besar untuk mengelakkan penumpuan pra-matang, dan mencegah zarah daripada terperangkap di dalam minimum tempatan. kami menguji prestasi *pdPSO* dengan IPM dan PSO menggunakan sembilan (9) fungsi penanda aras yang dinamik nya berbeza hasil eksperimen. Keputusan kami menunjukkan bahawa *pdPSO* menunjukkan prestasi yang lebih baik berbanding kedua-dua algoritma PSO asal dan kaedah IPM. Algoritma yang dicadangkan tidak mudah terdedah kepada penumpuan pra-matang, dan boleh mengendalikan perangkap minima tempatan dengan lebih baik berbanding dengan PSO asal. Oleh itu, hipotesis *pdPSO* mempunyai potensi untuk berfungsi dengan lebih baik berbanding variasi PSOs dapat

dilaksan akan. Kedua, kami berbanding kan prestasi *pdAPSO* dengan APSO, dan PSO menggunakan tujuh fungsi penanda aras. Algoritma yang dicadangkan terbukti mempunyai kapasiti yang besar untuk mengelakkan penumpuan pra-matang, dan mengatasi kutukan zarah terperangkap di dalam minimum tempatan. Akhir sekali, kami menggunakan algoritma hibrid tersebut (*pdPSO* dan *pdAPSO*) untuk menyelesaikan masalah pengelompakan di dalam robotik selara berkurumun. Hasil simulasi kami memerikan petunjuk yang jelas berkenaan keberkesanan algoritma yang dicadangkan. Algoritma hibrid adalah lebih baik dari segi ketepatan, kadar penumpuan, keunggulan, keteguhan dan keupayaan pengelompokan untuk set homogen robot sekumpulan berbanding dengan beberapa varian lain PSO. Di samping itu, algoritma kami juga digunakan untuk menyelesaikan masalah pembentukan corak robotik secara berkurumun. Kami juga membandingkan kinerja *pdAPSO* dan *pdPSO* dengan 9 negara dari algoritma PSO seni menggunakan 12 fungsi patokan. algoritma yang diusulkan kami memiliki mean keandalan 80,4% untuk *pdAPSO* dan 69,69% untuk *pdPSO*. Juga, *pdAPSO* dan *pdPSO* adalah kecepatan konvergensi yang lebih baik dibandingkan dengan 9 algoritma lainnya. Misalnya, pada fungsi Rosenbrock, FES rata-rata 8938, 6786, 10080, 9607, 11680, 9287, 23940, 6269 dan 6198 yang ditetapkan oleh PSO-LDIW, CLPSO, PPSA, PSOrank, OLPSO-G, ELPSO, APSO-VI , DNSPSO dan MSLPSO masing-masing untuk sampai ke optima global. Namun, *pdPSO* dan *pdAPSO* hanya menggunakan 2997 dan 2124 masing-masing yang menunjukkan bahwa *pdAPSO* adalah kecepatan konvergensi tercepat dan diikuti oleh *pdPSO*. Singkatnya, *pdPSO* dan *pdAPSO* menggunakan jumlah terendah FES untuk sampai pada solusi yang dapat diterima untuk semua 12 fungsi patokan.

"If I have seen further, it is by standing upon the shoulders of giants"

Sir Isaac Newton

To my parents, my wife Shola and children

ACKNOWLEDGEMENTS

I wish to express my profound gratitude to God Almighty for unflinching love, mercy, favour and provision, granted me to complete my PhD program, particularly this thesis work in this great citadel of learning. I acknowledge the immense and inspiring contribution of my dynamic, able, erudite and indefatigable supervisor, Dr. Effirul Ikhwan Ramlan for all the timely assistance and encouragement he gave me throughout this research work. Special thanks also go to the retired Prof. Sapiya Baba, Mr. Mazrul, and other academic and non-academic staffs in the faculty for their excellent tutoring. Benefiting from your rare wealth of teaching experience is a great privilege. I pray that the Almighty God will enlarge your coasts.

I have a deep sense of gratitude and indebtedness to the members and staffs of the Department of Computer Engineering, University of Maiduguri. I say a big thanks to the Vice Chancellor of Unimaid, the Registrar, Dean of faculty of Engineering, and the incumbent H.O.D of Computer Engineering Department, Unimaid. Also my thanks go to the Engr. Ismail and Dr. Aboaba for the help they have rendered to me in the cause of my study in UM. I also want to appreciate my colleagues in office, Mr. Bassey Steve Joseph, Mr Ali Baba and others that space will not permit me to mention their name. You are all helpers and lifters of destiny. See you at the top.

I will not fail to acknowledge the love, encouragement and support of my father in the Lord Pastor Ephraim Andy and Pastor (Mrs) Christy Ephraim. I also want to thank Pastor Sunday Olujimi and his wife, Pastor Tope, Pastor Okopedi, Pastor Mike, and every other person that have made my study in Malaysia a sweet experience. I thank all the worker and members of RCCG Province II, Malaysia. I will never forget the kindness demonstrated by Pastor Chris Iwenjora towards me. You will never lack helpers Sir.

I express my thanks to my friends and colleagues in Intelligent System lab. You are all unforgettable factors in my success at this PhD Programme. Finally, to my caring and loving wife, in person of Shola Deborah Dada, and my fruits: Daniel and David. Thanks for your understanding and support. I say a big thank you to my mother and siblings for their prayers and moral support. The Rewarder of men will reward you all beyond your imagination. Finally, I thank the management of University of Malaya for all the facilities and quality services they provided during my study in University of Malaya.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

PSO	Particle Swarm Optimisation
IPM	Interior Point Methods
PDIP	Primal-dual interior-point
pdPSO	Primal Dual Interior Point Method Particle Swarm Optimisation
pdAPSO	Primal Dual Asynchronous Particle Swarm Optimisation
APSO	Asynchronous Particle Swarm Optimisation
SPSO	Synchronous Particle Swarm Optimisation
I	Positive scaling factor
D	Position of the robot
D_t	Target of the robot
P (D, D_t)	Distance of the robot from the target
F_{att}	Attractive force
H	Non negative scaling factor
P(D, D_t)	Shortest distance between robot and obstacles
P_0	Compulsory distance of effect of the obstacles
GA	Genetic algorithm
BCGA	Bacteria Colony Growth Algorithm
RIN	Reactive Immune Network
AA	Ant Algorithm
ACO	Ant Colony Optimisation algorithm
dPSO	Distributed PSO

PRM	Probabilistic Roadmap Method
X_i	Position of PSO particle
V_i	Velocity of PSO particle
V_{max}	Maximum velocity of a particle
N	The number of particles in the swarm
i	The particle's number
w	PSO Inertia parameter
c_1	Cognitive scaling factor
C_2	Social scaling factors
$gbest_m$.	Global best of PSO particle
$pbest_{i,m}$	Personal best of PSO particle
X_i^d	i^{th} particle's d^{th} dimension's value
pop_size	Population size
gen_count	generation counter from 1 to max_gen
dimen	dimension
max_gen	maximum generations
EPSO	Extended Particle Swarm Optimisation
GDMEPSO	Group Decision Making Extended Particle Swarm Optimisation
WSN	Wireless Sensor Network
RSSI	Received signal strength indication
RSS	Rreceived signal strength
pePSO	Physically embedded PSO
dPSO	Distributed PSO
VL-ALPSO	Augmented Lagrangian PSO with Velocity Limits
DR PSO	Detection and responding PSO
CPSO	Charged PSO

ALPSO	Augmented Lagrangian Particle Swarm Optimisation
FIPS	Fully Informed Particle Swarm Optimisation Algorithm
RDPSO	Robotic Darwinian PSO
KKT	Karush-Kuhn-Tucker
SQP	Sequential Quadratic Programming
LP	Linear Programming
SOCP	Second-Order Cone Programming
SDP	Semi-Definite Programming
$\sum$	Primal-dual matrix (3.12),
τ	Scalar Value
S^{-1}	Scaling
P_s	Step vector
r_E and r_I.	Relaxation vectors
TOL	Halting tolerance
λ	The Lagrange multiplier
f(x)	The objective function
g(x)	Equality Constraint Violation
h(x)	Inequality Constraint Violation
BFGS	Broyden–Fletcher–Goldfarb–Shanno
Z1	Flocking zone 1
Z2	Flocking zone 2
Z3	Flocking zone 3
Z4	Flocking zone 4
C	Convergence centre
SP	Success Performance

X*	Theoretical global optimal solution
NFE	Average number of function evaluation required to find solution
SR	Success Rate

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Introduction

This thesis is a research work on the fusion of Particle Swarm Optimisation (PSO) and Interior Point Methods for Swarm Robotics. The proposed technique uses the Interior Point Methods to choose the initial feasible path for the robots in the search space. Then Particle Swarm Optimisation is used to find the optimal path for the robots in the swarm. Interior Point Methods provides the coordination between step computation and the bounds to make the robots navigate through the search space that the robot will navigate. This chapter introduces some background information on swarm robots; the research trend and motivation for the thesis, the problem statement, the aim and objectives of the research, the significance of the research; and the expected contribution of the research.

1.2 Background of the Research

The ability of a robot to locate the safest and shortest path is a very crucial factor to the success of any robotic system. The need to make the robot navigate through its environment without colliding with any obstacle during the course of finding its target is an important research area in robotics. The need to reduce computation cost, computation time, energy consumption, and delay in communication requires that more efficient and robust algorithms be developed. Several techniques have been developed to solve these problems. Examples of algorithms used for cooperative control of swarm robotic movement are the artificial potential field (Chanclou & Luciani, 1996), cell decomposition (Dušan, Mario & Mirjana, 2009), visibility graph (de Berg, Cheong , van Kreveld & Overmars, 2008), voronoi diagrams (O'Dunlaing & Yap, 1982), grid (Payton, Rosenblatt & Keirsey, 1993), genetic algorithm (Gao et al., 2008), fuzzy logic algorithm (Saboori, Menhaj & Karimi, 2006), and

neural network techniques (Yang & Luo 2004). Each of the techniques mentioned above has its own advantages and disadvantages. Conventionally, the major problems in swarm robots are the complexity of computation, the inability of the robots to escape from the local optima, and low ability of robots to adapt to the environment (Zheng, Jain, Koenig & Kempe, 2005).

The artificial potential force (APF) by (Khatib, 1968) is one of the few algorithms that have been used for real time robot applications. It provides a simple yet effective way to plan a path for a robot. Navigating the robot through the use a potential field implies that there is no prior knowledge of the pathway of the robot, neither was there any computation in advance. This enables the robot to automatically select the right way to reach their target (Dunias, 1996). Recently, the efficiency of the Artificial Potential Force has been increased by hybridizing it with some other computational methods. Miao (2009) reported that Simulated Annealing has been used together with APF to escape local minima.

Visibility Graphs are in the class of Deterministic Path planning methods. What is required to create a visibility graph is just linking two vertices of barriers, which are visible to each other, and linking the target and the robot to boundaries where they are completely visible to themselves. The merit of the visibility graphs is that there is assurance of locating the best path for any set of points in the graph (Russell & Norvig, 2010). The shortcoming of the visibility graphs is that they do not possess the ability to sustain their optimal performance when the maximum dimensions are high. Furthermore, they can only locate "semi-free" paths which are very close to barriers (Russell & Norvig, 2010). Moreover, the visibility graph algorithm requires that the robot has a very high accuracy in manipulating its way through the environment. The weakness of this algorithm is its low efficiency in searching for a path for the robot to navigate through (Yuan et al., 2004).

The genetic algorithm (GA), which was proposed by Holland in 1975, is a biology inspired search approach that uses a procedure similar to natural reproduction. In the work

of Beasley, Bull & Martin (1993), a GA has been used in the field of robotics for path planning or trajectory planning. The major drawback of GA is the unexpected deviations demonstrated in the iterations during the evaluation of the fitness function and generation of the offspring. However, such sudden leaps are not practicable for mobile robots (Yang, Ian, & David, 2007). Moreover, the genes of the GA can be trapped in local minima, meaning that there is no solution (Beasley, Bull & Martin, 1993). It has also been observed that the time taken to locate the first suitable path will be high if there are many obstacles in the environment. The response time of a GA during the optimisation process is usually very high thereby resulting in a slow convergence rate of GA genes as they search for the best path. Tang, Zhang, and Yang (2000) observed that low convergent probability in GA can result to low crossover probability and high mutation probability, which will affect the efficiency of the algorithm drastically. For instance, in GA, mutation operators are commonly used to serve the function of exploration, while cross-over operators assist the population to converge to good solutions. This is referred to as exploitation since the cross-over seeks to converge to a particular point in the search space. The mutation operator tries to prevent GA from early convergence so that it can explore other regions. It is preferable to explore more areas in the start of the search as this will guarantee exploration of more areas and increase the diversity in the population. Conversely, it is better to do more exploitations at the expiration of the search. This is to warrant the convergence of the population to the global optimum. In other words, when mutation rate is high, the search ability of GA is reduced. And when GA mutation rate is too small the search will most likely stop at a local optimum. Therefore, it is compulsory that algorithms that are more effective be developed to handle various swarm robotics tasks.

Since the introduction of Particle swarm optimisation (PSO), it has been extensively used for solving different swarm robotic tasks (Ellips & Davoud, 2010). PSO is a stochastic algorithm centered on population which is now being applied successfully in several fields like Electric Power Systems (AlRashidi & El-Hawary, 2009), water distribution network design (Montalvo et al, 2008), parameter optimisation in suspension system (Alfi & Fateh, 2010), resource allocation (Gong et al, 2012), task assignment (Ho et al, 2008), DNA sequence compression (Zhu, Zhou & Shi, 2011), e-education (Dascalu, 2011), and computational finance (Chiam, Tan, & Al-Mamun, 2009). Kennedy and Eberhart in 1995 provided the first PSO algorithm. They got their inspiration from the social behaviour of flock of bird, school of fish, and herd of animal (Kennedy & Eberhart, 1995). Just like was we have in other evolution algorithms such as GA, PSO possesses several advantages which include which makes it a desirable optimisation algorithm (Shi & Eberhart, 1998).

1.3 Research Trend and Motivation

I drew my inspiration to this research work on the fusion PSO and the Primal Dual method from the fact that Particle Swarm Optimisation (PSO) is one of the most popular algorithm in use because of its simplicity, ease of implementation and efficiency. It however exhibits some shortcoming, particularly its tendency to premature convergence and inability to escape from local minima. To address this weaknesses, many variants of PSO have been proposed in the literature. Some of these approach are centered mainly on manipulating some of the parameters used in the PSO algorithm optimisation process, whereas many others employed hybrid methods that integrate other optimisation algorithms with the standard PSO. It have been demonstrated in the literature that methods that hybridize PSO and some other optimisation algorithm have a better performance over the standard PSO algorithm.

The modern day Interior-Point algorithms have become recognised as the most ideal approach for solving large-scale linear problems (Laird, 2006). The Primal Dual method has been applied to convex optimisation problems where strong duality is required (Rockefeller, 1970). It has also been used for various nonlinear and non-smooth cost functions that are prevalent in network design, medical image reconstruction, and industrial engineering (Boyd & Vandenberghe, 2004). They can be easily parallelized which enables them to efficiently handle multi-dimensional problems (Bauschke & Combettes). The Primal dual method from literature can solve both linear and non-linear optimisation problems effectively (Laird, 2006).

We intend to develop a set of hybrid PSO algorithms that will be able to solve the aforementioned problems that are associated with PSO. The Primal Dual method, when integrated into PSO, will provide better balance between exploration and exploitation, preventing the particles from experiencing premature convergence and being trapped in local minima easily and so producing better results. The fusion of conventional PSO with Primal-Dual Interior-Point method will resolve the common issues associated with PSO algorithm and many of its variants. The integration will make our proposed system to have great capacity to prevent premature convergence, and prevent the particles from being stuck in the local minima.

In the past, many variants of PSO have been developed in the past to provide specific solutions for swarm robotics problem. For every swarm robotic problem investigated, a newly developed customized PSO is required to effectively solve the problem. For instance, (Hayes et al, 2003) and (Jatmiko et al, 2007) used PSO to provide solution to the odor localization problem of swarm robotics. Moreover, Hereford (2006) developed a distributed particle swarm optimisation algorithm to handle the scalability problem of swarm robotic system. Tang & Eberhard, (2011) proposed the Velocity Limit Augmented Lagrangian PSO

to solve the problem of collision with barriers in the search space by the robots in the swarm. Derr and Manic (2009) proposed the distributed PSO (dPSO) algorithm for navigation of robots in hazardous environment. Doctor 2004 applied PSO for path planning of unmanned vehicle. Zhang, Wu and Wang (2013) proposed the Chaotic PSO for avoiding collision with obstacles by swarm robots.

Hao and colleagues (2007) proposed a PSO algorithm that uses Polar coordinate system to move in a dynamic environment. Karimu (2012) proposed dynamic hybrid PSO algorithm to handle motion planning problem of mobile robot. Moreover, new variants of the PSO algorithm have been developed by fusing it with an already tested approaches which have been effectively used to solve complex optimisation problems. Academicians and researchers have improved the performance of PSO by integrating in it the basics of other famous methods. Some researchers have also made efforts to increase the performance of popular evolutionary algorithms such as Genetic Algorithm, Ant Colony and Differential Evolution, etc. by infusing the position and velocity update equations of the PSO. The purpose of the integration is to make PSO overcome some of its drawbacks like premature convergence, particles being trapped in the local minima, and partial optimisation.

1.4 Problem Statement

Though Particle Swarm Optimisation (PSO) is a widely accepted algorithm in different fields, it still suffers from common issues such as premature convergence, inability to effectively cope with a dynamic environment and failure of PSO particles to escape from being trapped in local minima. This provides possibilities for the development of new variants of PSO algorithms. Although successful in addressing those issues specific to a directed domain, these variants of PSO are still unable to resolve the issues effectively. The Interior-Point Methods (IPMs) are powerful tools for solving nonlinear optimisation problems. It has been depicted as the most robust algorithms for solving large-scale nonlinear

optimisation problems. On the other, similar to PSO, the methods are still plagued with several issues (e.g., how to handle non-convexity, the procedure for making the barrier constraint up to date is burdensome even with the presence of nonlinearities, and the necessity to guarantee progress in the direction of the solution). The particles in PSO are naturally inclined to fling to the infeasible areas from the feasible areas during the course of searching. This poses a threat to the searching efficiency of PSO. Many of the PSO variants are not really suitable for effective handling of swarm robotic tasks. The present variants of PSO cannot adequately handle the constraints and the complexities that characterise the dynamic environment. There is therefore the need to increase the efficiency of PSO algorithms for swarm robotic by developing new variants that can adequately handle complex constraints.

1.5 Research Questions

What are the problems associated with existing PSO algorithms that limit their suitability for swarm robotics tasks? What are the different ways by which PSO algorithms and Interior Point Methods can be applied to swarm robotics? What are the different ways by which we can bring together the advantages of PSO with the strengths of Interior Point Methods to overcome the weaknesses of PSO algorithms? What are the different ways by which the fusion of these two algorithms can be used to increase the performance of PSO and make them more suitable for handling swarm robotics cooperative movement? What are the processes that must be carried out for the new algorithms to perform (in terms of speed of convergence, escaping from been trapped in the local minima) better than the existing ones? What is the computational cost and computational time attached to using the new algorithms for swarm robotics? What are the ways of measuring the performance of the new algorithms (Best fitness, worst fitness, mean fitness, standard deviation of fitness)?

1.6 Aim and Objectives of the Research

To address the problem statement one, our objectives are: to carry out experiments using standard benchmarking functions on some variants of PSO algorithms that have find application in the field of swarm robotics. This will enable us to know the problems associated with the existing PSO algorithms that limits their usefulness for solving swarm robotics tasks. Moreover, we want to assess the convergence properties of the algorithms through these benchmark functions. And to validate the presence of premature convergence, inability of particles to escape being trapped in the local minima, and unsuitability of PSO for dynamic tasks in these PSO variants.

To tackle problem statement two, our objectives are: to propose new set of hybrid algorithms based on the integration of PSO and Primal Dual Interior Point Method, and use benchmarking functions to validate their performance with respect to the Best fitness, worst fitness, mean fitness, and standard deviation of the fitness for each of the algorithms. Also, we intend to test if the proposed algorithms have been able to overcome the problem of premature convergence and inability of particles to escape from local minima.

To deal with problem statement three, our objectives are: to apply our proposed algorithms to solve swarm robotic tasks such as aggregation, flocking and pattern formation to ensure the applicability aspect of the algorithm. It is essential not only to develop algorithms that perform better than its predecessors, but also applicable for the intended domain of cooperative swarm movement. These three objectives will ensure that our aim towards creating a singular (in PSO algorithm) and generic PSO derivative for all dynamic optimisation tasks can be achieved. Thus, eliminating the need to develop a new variant of PSO algorithm whenever a new dynamic optimisation problem is to be solved.

1.7 Significance of the Research

This thesis proposed the fusion of Particle Swarm Optimisation (PSO) and Interior Point Methods as cooperative movement control algorithm in swarm robotics. We developed the Primal-Dual Interior Point Particle Swarm Optimisation (*pdPSO*) algorithm to resolve the shortcomings of the standard PSO without the limitations of the IPM methods. We applied the Primal Dual to each particle in a finite number of iterations, and feed the PSO with the output of the Primal Dual procedure. We compared the performance of our new algorithm (*pdPSO*) with IPM and PSO using different dynamic benchmark functions. Our results revealed that *pdPSO* performed better than both independent PSO algorithm and IPM method. The novel algorithm is not susceptible to premature convergence, and can handle local minima avoidance better compared to conventional PSO, hence hypothetically has the potential to perform better than many variants PSOs. As part of our contribution to the field of swarm intelligence, we also developed the Primal-Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*) algorithm. We applied the Primal Dual to each particle in a finite number of iterations, and feed the APSO with the output of the Primal Dual. Also, our work demonstrated how the Primal Dual can be used to ensure better balance between exploration and exploitation, preventing the particles from experiencing premature convergence and been trapped in local minima easily and so producing better results. We compared the performance of our new algorithm (*pdAPSO*) with APSO, PSO, and Primal-Dual Particle Swarm Optimisation using 7 benchmark functions. Optimisation results reveal that *pdAPSO* offers similar or in many test cases better solutions than the other approaches used for the performance evaluation. Our proposed algorithm is shown to have the ability to avert premature convergence, and prevent the particles being trapped in the local minima which have characterized many variants of PSO.

Moreover, we applied our hybrid algorithms (*pdPSO* and *pdAPSO*) to adopt the

swarm robotics flocking motion problem. We hypothesize that the fusion of the two algorithms (PSO and Primal Dual) offers a robust prospect of preventing premature convergence of robots, and also make sure that the robots are not stuck in their local minima. The results of our simulation give a clear evidence of the efficacy of the algorithms.

These hybrid algorithms contributed to the field of swarm robotics by providing novel algorithms that perform better in terms of accuracy, convergence rate, stability, robustness and ability to flock for identical set of swarm robots. Lastly, this thesis also contributed to the field of swarm robotics by applying the new algorithms (*pdPSO* and *pdAPSO*) to solving the problems of pattern formation in cooperative movements of swarm robots.

1.8 Outline of Thesis

This thesis is organized into seven chapters. Chapter 1 provides an overview of the study. It outlines the general introduction, background of the research, research trend and motivation, statement of the problem, research questions, research aim and objectives, and the significance of the research work.

Chapter two discusses the characteristics of a swarm robotic system, the advantages and disadvantages of swarm robotics, detailed review of the different swarm robotic algorithms, and comparative review of foregoing research in the field of particle swarm optimisation. Moreover, we did an overview of the various Interior Point method algorithms. The primal-dual interior-point (PDIP) method, Barrier Method, Simplex Method, strengths and weaknesses of the Primal dual method, feasible and infeasible interior point method, line search interior point method, trust region interior point method were all discussed.

Chapter three presents a survey on the different variants of PSO and their performance on different benchmarking problems. We also seek to establish the grand truth about some

of these algorithms as discussed in literatures, and seek how they have been applied to swarm robotics problems.

Chapter four discusses our new hybrid algorithm named Primal Dual Interior Point Method Particle Swarm Optimisation (*pdPSO*). A comparative study of the new algorithm with the conventional PSO and Primal Dual method was done using nine benchmark functions. The chapter also presents another new hybrid algorithm named Primal Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*). We did a comparison of our new algorithm with the typical PSO, APSO and *pdPSO* using seven (7) benchmark functions. We also compared the performance of *pdAPSO* and *pdPSO.* The performance of *pdAPSO* and *pdPSO* was compared with 9 state of the art PSO algorithms using 12 benchmark functions.

Chapter five discusses the application of *pdPSO* and *pdAPSO* to flocking problem. We also applied *pdPSO* to solve pattern formation problems of swarm robotics.

Finally, chapter six summarizes the thesis. Our theoretical and practical contributions are discussed and the final conclusion is presented. In addition, we described the future research directions of this work.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This chapter gives a general review of state of the art swarm robotic algorithms. It also establishes the need for the study and also provides an in-depth look at some relevant literatures on algorithms implemented in the past and present and applied to swarm robotics tasks. Experiments were carried out to ascertain the strengths and weaknesses of some of the algorithms we read in literature, and also to highlight the drawbacks of Particle Swarm Optimisation (PSO) algorithms.

Sahin (2005) defined swarm robotics as the study of ways through which a vast number of simple inexpensive agents can be made to work together to bring about a preferred cooperative behaviour through communications at the local neighbourhoods among robots as well as between the robots and their surroundings. The attributes of a swarm robotic system includes firstly, a huge number of independent robots. Secondly, the ability to detect and transfer information from one robot to the other within the same local neighbourhood is very crucial in swarm robotics. Thirdly, they are decentralised and independent of global information. Lastly, they possess cooperative behaviour that can be attained via spontaneous formation of spatiotemporal structure, and communications among the robots and between the robot and their surroundings. Some of the cooperative behaviours in swarm robotics are discussed briefly in the section below.

According to Arkin (1998), there are several advantages and disadvantages of swarm robotic systems in comparison to a single robot. The advantages of swarm robotic system are as follows

i. Enhanced performance: breaking complex tasks into simpler ones and then making the robots to perform tasks cooperatively thereby increasing the efficiency.

ii. Job empowerment: some set of robots are authorised carry out specific jobs which are difficult for an individual robot to perform.

iii. Broad sensing: the radius of signal detection of a set of robots is broader than that of an individual robot.

iv. Parallel execution of jobs: a set of robots can set in motion several tasks in diverse places simultaneously.

v. Robust system: failure of one robot in the swarm does not mean that the system will fail or that the job will not be completed. There might be degradation in the performance of the system but it will complete its task even as some redundant robots in the swarm take up the challenge of completing the remaining tasks.

The shortcomings of swarm robotics are listed below:

i. Infringement: obstructions, collision, and intrusion can cause a set of robots to infringe on the movement of other robots.

ii. Doubt about the plans of other robots: directing a set of robots entails knowing the plans other robots in the group. Vagueness of purpose can result into contention rather than collaboration among the robots.

iii. Need for effective control algorithm: the fact that a large number of robots are involved means that there will be the need to develop an effective algorithm that will make the robots in the swarm to work in corporation to achieve their desired goal.

2.1.1 Cooperative behaviours

According to Brambilla et al. (2013), swarm robotics have many cooperative behaviours that can be used to solve sophisticated problems. These cooperative behaviours can be grouped into three principal sections:

i. Non-random arrangement behaviours: This deals with the ability of swarm robotic system to arrange and allocate robots and objects in the search space;

ii. Path finding behaviours: Have to do with the capacity of the swarm robotic system to plan and direct the way and manner in which the robots move in the swarm;

iii. Mutual decision-making and job distribution: Is the ability of the swarm to make a joint resolution or split the swarm into different clusters depending on their choice. A brief discussion about the different activities of the swarm robotic systems that requires joint decision are presented below.

2.1.1.1 Aggregation

This is a characteristics of swarm robotic system where all the robots in a swarm occupy the same locality in the environment and are clustered to accomplish a specific task (Camazine et al., 2001). Aggregation has been classified as the easiest cooperative behaviour in swarm robotics which is a requirement before other cooperative behaviours like flocking, foraging, and pattern formation (that involve the coming together of robot at the same place) can take place. Camazine et al. (2001), opined that the research on swarm robots' aggregation have been inspired by observing the behaviour of bees, birds, cockroaches, and fish. Some other approaches that are different from swarm intelligence methods have been employed to solve some of the extremely difficult forms of aggregation tasks in swarm robotics (Ferrante,

2013). In such cases, there is no available information, hints or set standards to decide where the robots should gather. It is required that the robots gather themselves together at a haphazard location. Some researchers such as Minsky (1967) have used likelihood finite-state automaton to do aggregation. Also, Nolfi and Floreano (2004) employed artificial evolution to achieve aggregation. These approaches that have been mentioned have their limitations as some of the robots can still be trapped thereby not converging with other robots in their destination (Ferrante, 2013).

2.1.1.2 Pattern formation

Pattern formation basically involves changing the position of each robot in the swarm based on some predefined rules in order to form a particular shape (Yukiko, 2013). This is also a non-regular arrangement behaviour that can be utilized to arrange robots in an ordered and rhythmic style. According to Meinhardt (1982), the inspiration of designing algorithms to make swarm of robots to form a particular pattern came from natural processes like development of colour patterns in animals. Another example as stated by Langer (1980) is physical activity such as mineral development.

Egerstedt and Hu (2001) proposed a pattern formation method that uses the organization approach for a swarm of robots to form a given shape. The technique was used in a simulated environment that takes obstacle avoidance into consideration to control the movement of robots to form a triangular shape. The strength of this approach is that the tracking of robots was properly done thereby stabilizing the pattern formation error.

Koo and Shahruz (2001) proposed another pattern formation approach that used a make a set of unmanned aerial vehicles (UAVs) to form a preferred shape using centralized

pattern formation method. The main emphasis of their research work is the computation of the Route through which the unmanned aerial vehicles will travel through.

Belta and Kumar (2002) proposed a method that uses an invariable kinetic energy measurement to create plane paths for a swarm of robots to navigate. The closeness between one robot and other robots in the swarm can be regulated using some parameters. However their approach failed to consider obstacle avoidance and it is not scalable. Krishnanand & Ghose (2005) proposed a pattern formations algorithm of simple robots using indigenous prototypes and attitudinally distributed communications. Ikemoto et. al. (2005) presented a pattern formation approach based on steady spatial formation of similar robots into a desired shape. Chen, et. al. (2008) also proposed a decentralized pattern formation algorithm for making mobile robots to form a given shape. Elor and Bruckstein (2011) proposed a pattern formation approach that deploys multiple identical robots in a swarm to form a particular shape.

From the research work that we have considered so far, there is no guarantee that the robot will converge to form the desired pattern (Gautam & Mohan, 2013). If the given shape is formed, it is a weak formation that has no firm symmetrical shape. Many of the determining factors about the formation of the patterns are based on presumptions. Examples of such deductions include the detection, direction-finding, interaction and computational competences of the robots. The robots also have negligible or no perceptive of other robots in the search space. Also, there is insignificant or no interaction among the robots (Gautam & Mohan, 2013).

2.1.1.3 Cooperative exploration

This is an explorative behaviour utilized by robots to search the environment where they are positioned, locate their target(s), and successfully move to their destination (Camazine et al., 2001). As reported by Camazine et al. (2001), the origin of inspiration for cooperative exploration is from insects like ants, termite, and bees. Howard et al. (2002), proposed a swarm robotic cooperative exploration technique that uses a potential-field-based method to deploy mobile robots in a swarm. The fields are fabricated in a way that each node is prevented by both barriers and other nodes, thus compelling the network to extent its area of coverage in the environment. Their method is both decentralised and scalable. Their experiment is however incomplete because they did not take into consideration some external and internal factors such as network size, environment, original situations, weights, size of the node, and viscidness.

Ducatelle et al., (2011) proposed a swarm robotic cooperative exploration technique that allows the robots to direct one another's movement by passing their path finding information by means of wireless network created in the swarm. They conducted their experiments under two different situations. Firstly, the swarm directs a robot to its destination. Secondly, all the robots in the swarm move to and fro between two destinations. The method was found to be efficient and robust in the two situations. It is robust to failures of robots in the swarm. The approach helps the robots to organization themselves thus increasing the efficiency of exploration. The approach also possesses the ability to locate the shortest routes in chaotic environments.

2.1.1.4 Mutual decision-making and job distribution

This deals with the ability of robots in the swarm to inspire one another during decision making (Camazine et al., 2001). The mutual decision making can either result into reaching a compromise to accept the joint decision or selection from the list of feasible options. Instances of mutual decision-making are derived from animals. For example, the idea of swarm robotics sprung from the study of the behaviour of cockroaches (Amé et al., 2006). Demonstrations of job distribution are very common in animals that live in castes like ants and bees (Camazine et al., 2001). Various techniques have been applied for joint decision-making as explained by Campo (2011). Examples of approached that are used for mutual decision making are likelihood finite-state automaton (Garnier et al., 2005), and statistical-physics (Montes de Oca et al., 2011). Likelihood finite-state automaton that was proposed by Bonabeau et al. (1997) in their seminar work is used for job distribution in swarm robotics (Liu et al., 2007; Brutschy et al., 2012).

Liu et al., (2007) presented a research work that uses a naive acclimatization system that spontaneously change the proportion of robots in a swarm that are involved in foraging and those that are resting. The advantage is that energy is conserved. The robots in the swarm have some set of rules that enables them to effectively search for food, avoid colliding with other robots in the swarm during search, and also ensure that they retrieve their food. The robots also help their team mates to successfully retrieve their own food as well. The job of searching for food is distributed among the foraging robots and the resting robots. The results of their simulation demonstrates that mutual decision and job distribution is achievable in swarm robots. Their work further revealed that mutual decision is highly necessary when there is shortage of food.

Moreover, the swarm acclimatization system helps in optimisation of the energy consumed regardless of the poor perception capacity and inadequate information transfer of the robots to the other robots around them. Another strength of their work is the ability of the swarm to detect and adjust to changes in the environment.

2.2 Advances in Swarm Robotic Algorithms

Many researchers have in the past used different approaches for the task of planning swarm robot's mobility in the search space.

2.2.1 Artificial Potential Fields (APF)

One of the methods used for controlling the navigation of robots in a swarm is the artificial potential field (APF). It was proposed by Khatib in 1968. It is one of the few motion planning methods for real time robot applications. It provides a straightforward but efficient way to plan a path for a robot. The APF is a function whose slope angle is used to calculate a force applied to the robot. This force compels the robot to navigate to its target without being stopped by obstacles. APF is made up of force vectors, which can be as a result of the obstacles or the location of the targets in the search space. Depending on the state of the robot with respect to the environment, the forces acting on the robot can either be repulsive, attractive or random. The equation D = [d, q] can be taken as the position and direction of vector. The attractive potential field force acting can be written as:

$$U_{att}(D) = IP^2(D, D_t), \text{ where}$$

I: is a positive scaling factor

D: is the position of the robot and

D_t: is the target of the robot

$P(D, D_t) = \| D_t - D \|$: is the distance of the robot from the target

F_{att} : is the attractive force (it is a negative gradient of the attractive potential field function which is less than zero)

This means that

$$F_{att} = - \nabla [U_{att}(D)] = IP(D, D_t)$$

The repulsive potential field function is described by

$$U_{att}(D) = \begin{cases} 0.5\eta \left(\frac{1}{p(d,d_0)} - \frac{1}{Po}\right) & P(d,d_0) \le Po \\ 0 & P(D,D_0) > Po \end{cases}$$ where

η: is non negative scaling factor

$P(D, D_t)$: is the distance that is the shortest between the robot and obstacles

P_0: is the distance between the robot and the obstacles

P_0 is a constant value, which depends on the target of the robot, the type and shape of obstacle. It is usually less than half the extent of space between the obstructions or shortest length from the obstacles to the goal.

The repulsive force that is applied whenever the robot is yet to get to the target is:

$$F_{rep} = - \nabla [U_{rep}(D)]$$

$$= \begin{cases} \eta \left(\frac{1}{p(d,d_0)} - \frac{1}{Po}\right) \frac{1}{P^2(d,d_0)} & P(d,d_0) \le Po \\ 0 & P(D,D_0) > Po \end{cases}$$

The resulting force is $F = F_{att} + F_{rep}$

According to Cao, Huan, and Zhou (2006), the force F controls the navigation of the robot from the start to the goal. Below is an illustration of how the potential force works.

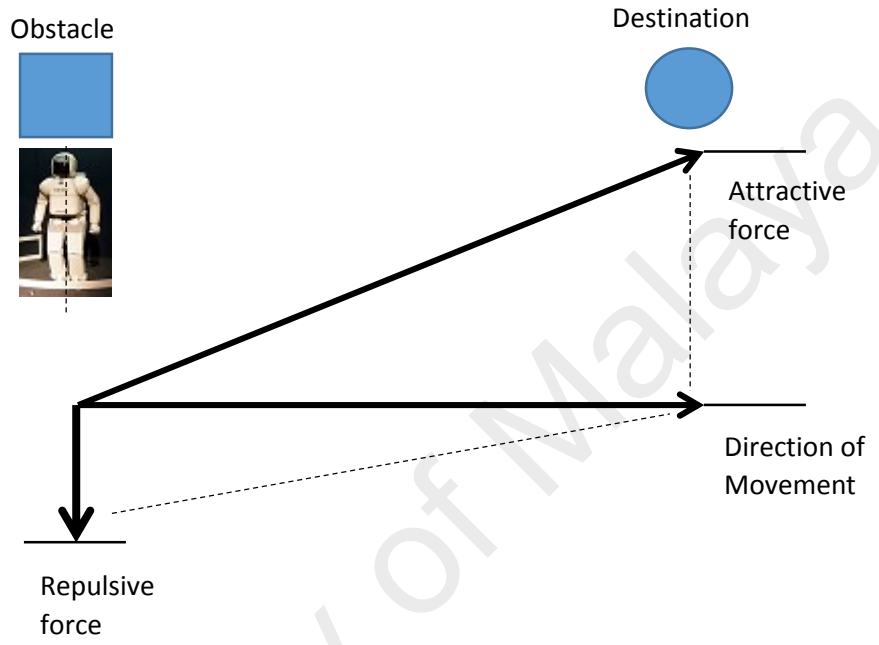


Figure 2.1: Potential Field (Diagrammatical representation of the APF showing how the attractive and repulsive forces compel the robots to navigate from its starting point avoiding obstacles on its way and getting to its destination).

Some of the desirable qualities of the APF include its simplicity of use, efficient mathematical evaluation, real time usage, applicable to both redundant and non-redundant robots, and the incorporation of the dynamics of the robot (Ren, et al., 2006). The major drawback of APF is the local minima which could lead the robot been trapped before reaching its goal. Other weaknesses of APF according to (Wang et al., 2013) are oscillation, and Goal Non Reachable with Obstacle Nearby (GNRON). Also failure is possible even when a valid path exists and failure modes are visible to onlookers. Navigating the robot through the use a potential field implies that there is no prior knowledge of the pathway of the robot, neither

was there any computation in advance. That implies that the robot 'chooses' automatically its way to reach its goal (Dunias, 1996).

Nowadays, the Artificial Potential Force is been hybridized with other computational methods to improve its efficiency (Lie Tang et al., 2010) and Miao (2009)). Wang (2000) used the Road Map approach to plan path for mobile robot. The shortcoming of the method is that it cannot be used for complex environments due to the high volume of computations that is involved. Latombe (1990) used the Cell Decomposition method for path planning of robot. Zhu, Yan, and Xing (2006) integrated Artificial Potential Field into Simulated Annealing for planning the path of mobile robot. They were able to solve the problem of the robot been trapped in the local minima. Boschian and Pruski (1993) applied the Grid method to robot path planning. The technique is however ineffective because it cannot solve the problem of been trapped in the local minimum. Visibility graph was used by Li, Ye and Tan (2002) for robot path planning. Swarm intelligence and Potential flow was used by Hu, Wu, and Wang (2007) to navigate mobile robot in both static and dynamic environment. The problem with this approach is that unless the obstacle is conceptualized as a normal circle having radius, it is difficult for the robot to move from one point to another in the search space. Simulated Annealing was used by Hui (2009) to plan the path for robot in a dynamic environment.

2.2.2 A* Algorithm

Another algorithm that has been used for swarm robotics is the A* algorithm. The A* (pronounced A star) was first described early in 1968 by Hart, Nilsson and Raphael (Chestrutt & Lau, 2005). The algorithm is a best-first, tree search algorithm, and able to find the shortest route for a robot to travel from the initial spot to the target. A modified A* algorithm that was applied to a real robot was developed by Chestrutt & Lau (2005) to calculate path for ASIMO (a robot with human characteristics). They engaged a framework of crisscrossed

cells to represent the environment while colour cells were used to represent the obstacles. The region that has no gravitational or electromagnetic fields is represented by some bits. The distance to be covered by step positions that will make the robot to navigate towards the target position is computed by the algorithm. The algorithm also includes information on known static and moving objects with anticipated paths. There are three cost functions been used by the A* algorithm. The cost functions help the algorithm to constrain the step nodes used by the robot. The cost functions include: one, costing the step locality with respect to the surroundings to resolve whether the location is safe or not. Two, calculating the attached cost if robot is to take the step desired. And three, using a typical mobile robot organizer to provide a navigational solution so as to determine the estimated remaining cost. This is used to prevent the robots from been caught in the local minima. Zeng, Zhang, and Wei (2012) proposed a Genetic Algorithm-based global path planning for mobile robot using A* Algorithm. The approach used the MAKLINK graph theory and Dijkstra algorithm to create unrestricted space prototype of robots and find best collision free path for the robot respectively. A* algorithm and genetic algorithm are then used to generate the global optimal path of the robots.

2.2.3 D* (Dynamic A*) Algorithm

Similar to the A* also is the D* algorithm. D* (pronounced "D star") is an incremental search algorithms that uses graph representing cost. The basic D* method was originally proposed in 1994 by Stentz (1995). He described the D* as a method used in planning direction of mobility in non-static new environments. There is a great reduction in the computational cost as the robot strategies for the best target coordinate by computing the best routes leading to the robot's destination. Depending on the new information gathered from the environment by the robot, it will replan new paths and produce a new path for the robot

to navigate. There is no need for D* to do a complete replannnig of the navigation direction all the time when it receives any new information. It updates the path cost locally when environment changes, and tries to get the best global path when possible. To improve the efficiency of the algorithm, some varieties of D* have been proposed such as the Focused D*, Framed-quadatree D* and Field D*. For example, the Focused D* algorithm have been applied in real-time path replannnig. The algorithm calculates an original path through which the robot will navigate from the initial state while effectively adjusting its path during the navigation whenever there is fluctuations in the arc costs. One of the strength of the Focused D* algorithm is that it is able to generate an ideal path for the robot to navigate to its target. The performance of the Focused D* is better than that of the original D* algorithm. The integration of an exploratory Focusing function to D* makes it a generalized form of A* that can copy in a non-static environment (Stentz, 1995).

2.2.4 Genetic Algorithm (GA)

Genetic algorithm was proposed in 1962 by Holland (Holland, 1993). GAs are a set of adaptive techniques which provide solution to the problems of search and optimisation involving substantial exploration of search landscape (Mantas & Andrius, 2007). They make use of natural process of choosing the most suitable chromosome in the population to proffer solution to optimisation problem (Mantas & Andrius, 2007).

Genetic algorithms can also be defined as biology inspired search approach that uses a procedure similar in nature to evolution discovered by Charles Darwin (Goldberg & Holland, 1988). According to Goldberg (1994), GAs have been extensively accepted as an algorithm that can solve complex problems that traditional algorithms cannot provide adequate answer to within a realistic time limit (Goldberg, 1994). A primary population of individuals is arbitrarily created, and every single individual is evaluated. After the

evaluation, many individuals with higher fitness are generated and the ones with lower fitness are throw out of the population. The GA uses genetic operators such as mutation, crossover and natural selection to produce better generation ((Forrest & Mitchell, 1993) and (Keisam, 2014)). The process discussed here follows the natural biological process by which changes occur in a chromosome through crossover and haphazard mutation (Trivedi, Lai & Zhang, 2001). In the work of David, Bull & Martin (1993), genetic algorithm has been used in the field of robotics for path planning or trajectory planning. Ghorbani (2009) employed Genetic Algorithm for the navigation of mobile robot. The path of robot was planned in a static environment by (Sugihara & Smith, 1997) and (Gallardo et al, 1998) using Fixed Length Binary Strings Genetic Algorithm. The shortcoming of this approach is that it takes longer time to generate solutions because chromosomes that have fixed length are not appropriate for complicated environment. Another approach called Binary Coded Genetic Algorithm where the gene indicates the subsequent bearing and distance was proposed by Tu and Yang (2003). The deficiency of this approach is that it leads to false solutions like paths that may not get to the end of the search space (or boundary condition). An evolutionary planner that uses simple genotype to depict suitable paths for robot to navigate was proposed by Xiao et al (1997). A modified visibility based method and repair operator that generate suitable paths which are then exposed to evolutionary technique was presented by Dozier et al (1997). However, the approach can only create path for robot to avoid convex obstacles, while it finds it very hard to proffer solution to obstacles that are concave in type. More so, the computation time is high as it takes longer time to convert chromosomes to phenotypes. According to Ripon, Kwong and Man (2007), the binary nature of the repair operator results in decline in accuracy as local tuning of solution is problematic. Most of the existing evolutionary based path planning algorithms, the computation load and execution time increases as the population increases.

There are some drawbacks with the previously proposed genetic algorithms which makes them unsuitable for swarm robotics. Firstly, genetic algorithm during optimisation usually produces unexpected deviations and such sudden leaps are not practicable for mobile robots. Secondly, premature convergence that makes the population to converge untimely to non-optimal local minima is prevalent in GA (Davis, 1991). Thirdly, there are some illusory functions that are 'hard' for most GAs to solve. Fourthly, GA finds it hard to cope with noisy functions (Goldberg, Deb, and Clark (1992), and 1993). Finally, optimizing many conditions always create problems for GA (Fonseca and Fleming, 1995). Some recent variants of GA proposed by Ai & Wang (2011) have made some improvement in the performance of GA in overcoming the problem of premature convergence that is inherent in the current GAs. One of their hybrid algorithm fuses GA with the downhill simplex method, while the other one integrate the conjugate gradient method into GA. Below is a flowchart of the genetic algorithm.

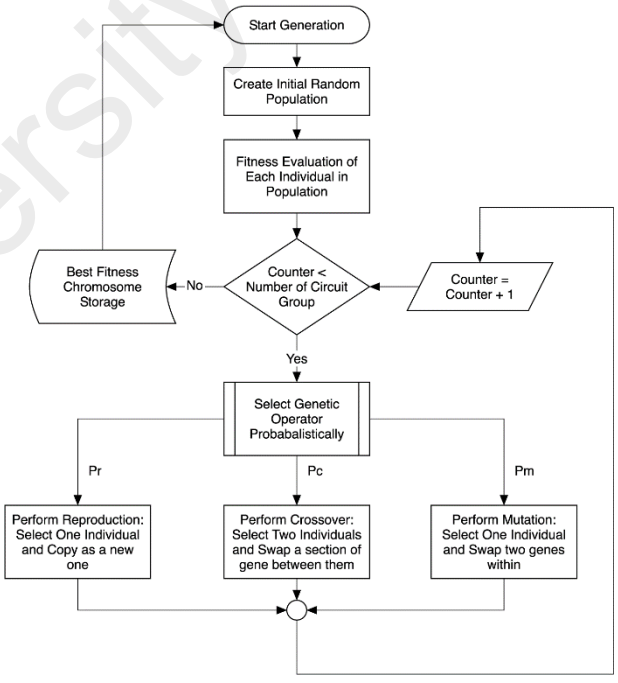


Figure 2.2: Flowchart of the Genetic Algorithm (Adapted from Chan, Lau, & Ko, 2000)

2.2.5 Bacterial Foraging Optimisation Algorithm

Bacteria Foraging Optimisation Algorithm (BFOA) was presented by Passino in 2002 (Das et al., 2009). It is an emerging algorithm in the class of bio-inspired optimisation algorithms. Foraging is a characteristic exhibited by group of bacteria and not an individualistic character. This algorithm possesses the attributes of the bacterial foraging forms such as chemo taxis, digestion, procreation and detection of quorum (Binitha & Sathya, 2012). Chemo taxis involves movement from cell to cell. It is the assembling of bacteria to areas where there is abundance of nutrients in an unconstrained manner. This establishes a process that enables transfer of information from one cell to another. Chemo taxis mimics the biological behaviour of how bacteria move when looking for nutrients. This is also called swim/tumble ((Binitha & Sathya, 2012) and (Das et al., 2009)).

Procreation is achieved in BFOA through the process of natural selection. It is only the bacteria that seem to best adapt to the procedure that continue to exist, and transfer their genetic attributes to subsequent generations. The less adapted bacteria do not survive this phase. To enhance the global search capability of the algorithm and also avoid being trapped in local optima, the exclusion-spreading mechanism is incorporated into the algorithm (Binitha & Sathya, 2012). This strategy randomly chooses the sections of the bacteria population to reduce and scatter into arbitrary locations in the environment.

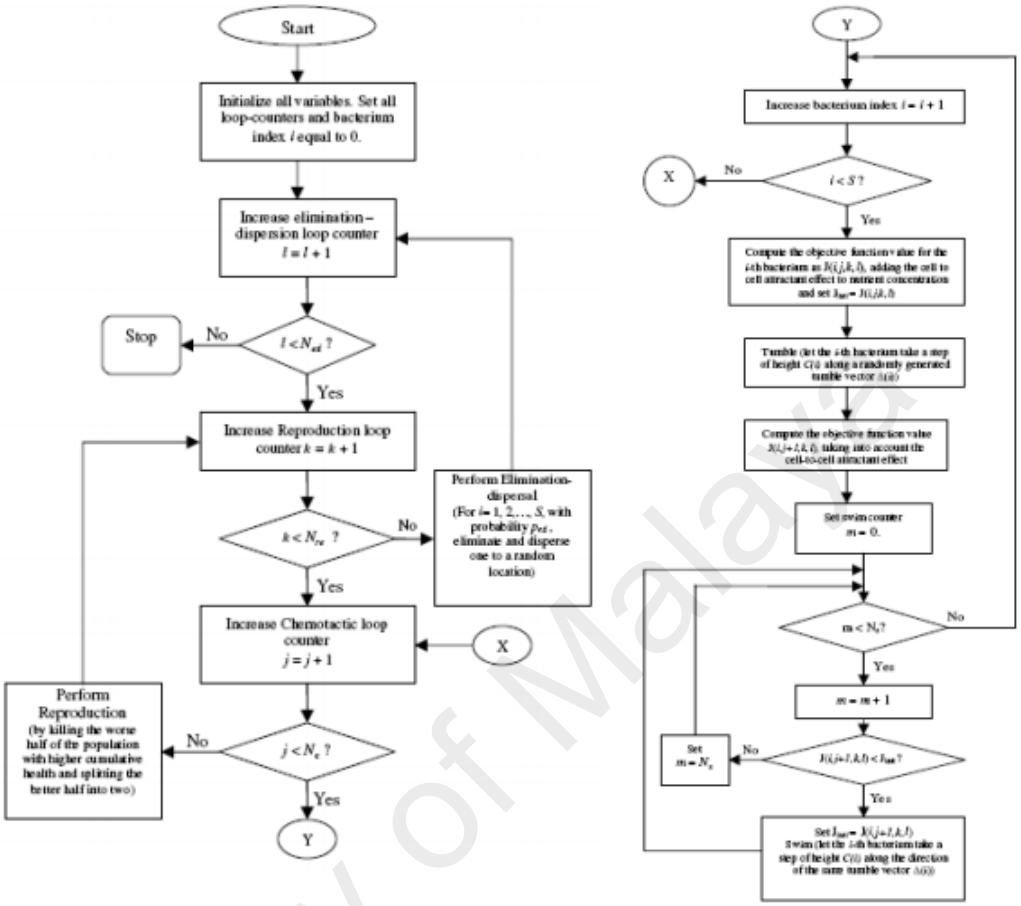


Figure 2.3: Flowchart of the Bacteria Foraging Optimisation Algorithm (Adapted from Das et al., 2009)

2.2.6 Artificial Bee Colony (ABC) Algorithm

Karaboga and Basturk (2007) proposed the Artificial Bee Colony algorithm. It was inspired through the study of the foraging activities of a swarm of bee. There are three (3) categories of artificial bees in the ABC algorithm: employed bees, onlookers and scout. A bee that is hanging around the dance area to decide on the origin of food to choose from is referred to as onlooker. The one that is going to the origin of the food inspected before by it is called employed bee. The last type of bee in ABC algorithm is called the scout bee. It conducts arbitrary exploration for finding new origins of food. The location where the origin

of the food is denotes a potential solution to the optimisation problem (Binitha & Sathya, 2012). Also, the quantity of sugary fluid called nectar available at the food origin is directly related to the fitness of the obtained solution. Based on this principle, the ABC algorithm creates a swarm of simulated bees and began to move arbitrarily in the search space. Whenever the bees locate nectar in the targeted location, there is communication. The optimal solution to the problem is obtained from the massive communication among the bees (Binitha & Sathya, 2012).

At the beginning a set of solutions (s_1 = 1, 2… N) that is randomly dispersed across the N problem space. A hired bee generates an amended solution in her memory. The solution is based on the information she is having. It then tests the fitness value new solution (nectar). Given that the nectar in the earlier one is less than that of the former one, the bee will record in its memory the updated position and ignores the old one. All the hired bees share the sugary information of the origin of food and their position information with the onlooker bees as soon as they are through with their search process (Binitha & Sathya, 2012).

The subsequent stage is Procreation. This depends on the value of likelihood connected with the origin of the food. The likelihood is represented as L_i. The origin of the food is selected by the simulated onlooker bee.

$$L_x = \frac{fitness}{\sum_{n=x}^{N} fitness_n}$$

Where, N is the number of origins of food (the number of hired bees), fitness is the fitness of the solution x which is relative to the amount of the sugary fluid in the origin of the food at the location. The final stage is called Substitution of bee and Choice making. The location of food is abandoned when the position cannot be enhanced by a prearranged number of circular

movements. The value of prearranged number of circular movements is a very vital constraint factor of the ABC algorithm. It is referred to as the threshold of abandonment (Binitha & Sathya, 2012). The simulated bee generates and evaluates the location of each bee. It then compared its performance with that of its previous one. The previous food origin is replaced with a new one in the memory of the quantity of sugary fluid in the new food is equal or better than that of the previous origin; else the previous one is kept in the memory.

2.2.7 Ant Algorithm (AA)

The Ant Algorithm was proposed by Dorigo, Maniezzo, and Colorni (1996). It draws its inspiration from the foraging behaviour of ants. The term referred to as stigmergy which was initiated by Grasse in 1959 (Binitha & Sathya, 2012). Stigmergy can be described as the oblique transfer of information among a self-arranging evolving system through parties that are changing their surroundings. The most fascinating part of the cooperative behavior of various species of ant is their capability to locate shortest routes between the ants' habitual resort and the food origins by trailing pheromone traces. Afterward, the ants select the route to follow by means of a probability method to make choices depending on the volume of pheromone deposited by the ant. The greater the pheromone's trails, the more attractive it is. Since ants leave behind pheromone on the route they are trailing, this behaviour gives rise to a self-strengthening process resulting in the creation of routes indicated by high pheromone intensity. Through modeling and simulation of ant foraging behaviour such as brood sorting, nest building and self-assembling, algorithms can be created that may possibly be employed to solve difficult combinatorial optimisation problems.

The performance of the algorithm was verified using common benchmark Travelling Salesman Problem (Binitha & Sathya, 2012). Ant algorithm prototypes the activities of physical ants that possess the capacity to find the swiftest path from one food origin to the

successive one without any obvious sign (Gengqian et al, 2005). They also possess the ability to adapt fast to any environment change that may need exploring another route if they are confronted with any obstacle in the old one. This is because the hormone that the ants deposit called pheromone has an effect on the remaining ants' choice of path. When many ants chose a path the quantity of the chemical secreted on the path increases, thereby serving as attractant to other ants of the same species to choose that same path. The pheromone will fade away as time goes on (Gengqian et al, 2005). The Ant algorithm uses the model of group ant to navigate the swarm robot. The weakness of this approach is that it is complicated as it needs a lot of computational time to arrive at the solution (Gengqian et al, 2005). Some of these weaknesses have been dealt with in the work of Dorigo and Stützle (2004) who proposed a MAX-MIN ant algorithm that adjust the pheromone to hinder the search from being trapped in local optimum.

The Ant Colony Optimisation (ACO) which is another metaheuristic algorithm was designed to provide good enough solution to optimisation problem was proposed by Dorigo and Di Caro (1999). The aim is to simplify the general technique of providing solution to combinatorial problems by approximate solutions built on the universal behaviour of physical ants. According to Binitha & Sathya (2012), ACO have three main steps which are arranged as functions. They are:

1. Create Solutions for the ant – The simulated ants navigate through neighbouring positions of a problem using an evolution rule to repeatedly generate solutions.
2. Perform the updating of pheromone - This deal with providing the update of the pheromone traces after creating comprehensive solutions. The ACO algorithm also put into consideration a situation when the traces of pheromone vaporize with time. Vaporization of pheromone traces assists ants only remember the good solutions that were created during the course of running the algorithm.

3. Perform deamon actions – This is an alternative step whereby the algorithm makes use of extra updates from overall viewpoint. This step can also involve using extra pheromone to boost the ability of the algorithm to get the most ideal solution produced.

ACO have been applied to the field of swarm robotics. For instance, Brand et al (2010) applied the ant colony optimisation (ACO) algorithm to find the fastest and obstacle free route for robot motion planning in a network. In their experiment obstacles with varying structures and magnitudes were simulated in an environment that is dynamic. The result of the computer simulations showed that for the new network, the best path can be re-routed successfully by ACO algorithm after the addition of obstacles. The weakness of the two approaches discussed above is that there is a high computational cost incurred. This is because much time is required for the ants to deposit their pheromone, update it, and locate the origin of the food. Ant Colony Algorithm (ACO) was used by used by Sugawara (2004) to provide solution to the navigation problem of robot in a virtual dynamic environment.

2.2.8 Artificial Neural Network

The idea of neural networks (also known as artificial neural networks) was motivated by the recognition mechanism of the human brain (Ben & Patrick, 1996). The human brain can be described as a complicated, multidimensional parallel processing computer. Our digital computer is not as fast in performing computations as human neurons (Simon, 1998). According to Simon (1998), human vision is a good example for understanding this difference. Neural networks are a form of multiprocessor computer system with many simple processors called neurons. There is a high level of interconnection, simple organized messages and communication that are exchanged between these neurons. These neurons

usually have a limited memory. The neurons are saddled with the responsibility of receiving inputs from other neurons within the network or importing input from outside sources. These inputs are used to calculate the output for the neural network and can also be sent as input to the neurons of the next layer. The communication routes (also known as weights) transport the computed data. The weights which establish communication between two neurons have definite values and which can be timed when training the network. The weights are tuned in parallel, implying that the computations by many neurons can be processed concurrently. The learning rule is used in training the data and the extent of tuning of the neurons is subject to the training data.

For example, if a neural network multiple layers. The layers are arranged as follows; leftmost layer is the input layer; it accepts data from a source. The rightmost layer two is the output layer that transports the calculated data from layer one out of the neural network. The other layers are referred to as the hidden layers. It's input and output signals continue in the neural network. The neural network above is a case where each of the neurons in one layer is totally connected with all neurons in the previous layer and so on. For any given circumstance, a neural network is expected to figure out the most accurate results. With the application of the training rules and the weights which can be changed, the neurons in the neural networks can be used in creating a collision free path that the robot will navigate in the search space. The advantage of using neural network is that it is computationally simple (Simon & Chaomin, 2004).

Artificial Neural Network was used by Qu (2009) for navigation and obstacle avoidance of mobile robots in dynamic environment. Du, Chen, and Gu (2005) proposed a mobile robot global path planning approach for a static environment using Neural network and genetic algorithm. A new path planning method based on neural network with obstacle avoidance that enables cleaning robots in dynamic environment to extend its search to the

nooks and crannies of the search space was proposed by Simon and Chaomin (2004). A multilayer Artificial Neural Network model was proposed was proposed by Youssef (2012) for rover systems to enable them carry out self-directed path-planning to favourably move towards their targets in difficult terrains while avoiding obstacles.

2.3 Particle Swarm Optimisation (PSO) Algorithms

Different variants of PSO algorithms have been proposed in the literature. For example, Robinson et al. (2002) developed the GA-PSO and PSO-GA and used them to solve a specific electromagnetic application problem of projection antenna. The results of their experiments revealed that the PSO-GA hybrid algorithm performs better than the GA-PSO, standard PSO only and GA only. He proposed the hybridization of GA and Hill Climbing algorithm the same year and used it to solve unconstrained global optimisation problems (Krink & Løvbjerg, 2002). Conradie, Miikkulainen, and Aldrich (2002), developed the symbiotic neuro memetic evolution (SMNE) algorithm when they hybridized PSO and 'symbiotic genetic algorithm' and used it for neural network control devices in a corroboration learning context. Grimaldi et al. (2004) developed the genetic swarm optimisation (GSO) by hybridizing PSO and GA. They later went ahead and used their algorithm to solve combinatorial optimisation problems. They presented different hybridization approaches (Gandelli, 2007). They authenticated the genuineness of GSO using different multimodal benchmark problems and applied it in different domain as demonstrated in Gandelli et al. (2005), Grimaccia et al. (2007) and Gandelli et al. (2006).

Juang (2004) proposed the HGAPSO by hybridizing PSO and GA. Settles and Soule (2005) proposed the BS algorithm which unites the velocity and position update equation of PSO and the concepts of selection, crossover and mutation of GA with a supplementary parameter referred to as the breeding ratio to decide on the percentage of the population that

will go through breeding in the present generation. Jian and Chen (2006) proposed the PSO-RDL which is a PSO- GA hybrid (where the GA makes use of recombination operator and dynamic linkage discovery). This algorithm was used to provide solution to 25 unconstrained test problems having different degrees of sophistication. A new hybrid algorithm in where PSO was used to produce the first population for GA was proposed by Mohammadi and Jazaeri (2007). This algorithm was applied to proffer solution to an IEEE 68 bus problem. Esmin et al. (2006) introduced the HPSOM algorithm which is the hybrid of PSO and a GA mutation operator. The algorithm was used to solve unconstrained global optimisation problems. Kim (2006) presented another GA-PSO algorithm that uses the PSO and the concept of Euclidean distance. He applied it in getting the local and global optima of Foxhole function.

Yang et al. (2007) proposed the PSO-GA centered hybrid evolutionary algorithm (HEA) that splits the evolution process into two phases. Kao and Zahara (2008) introduced another hybrid algorithm named GA-PSO. This algorithm was used to solve 17 unconstrained multimodal test functions. Ru and Jianhua (2008) proposed a hybrid of GA and PSO that combines the strengths of PSO and GA and also uses the idea of breeding individuals originally exhibited by GA with the idea of self- enhancement of PSO, where the particles improve themselves as a result of the social and cognitive scaling factors. The hybrid particle swarm optimisation (HPSO) was introduced by Shunmugalatha and Slochanal (2008). This algorithm absorbs the process of propagation and subpopulation in GA into PSO. The result of their experiment shows that HPSO's convergence rate is high and the solution is better. Li et al (2008) proposed a PHGA which is a hybrid of PSO and GA. The algorithm employed enhanced genetic methods such as the nonlinear choice of position (Mahanti & Chakrabarty, 2007) used in GA method.

Ting et al. (2008) proposed a hybrid constricted algorithm named GA/PSO used for

providing solution to challenges of load flow control. A hybrid GA/PSO algorithm was presented by Jeong et al. (2009). This algorithm was used to give solution to multiobjective optimisation problems. Valdez et al. (2009) reported a PSO + GA hybrid algorithm. It works by using GA + PSO fuzzy rules to determine if it will select GA particles or PSO particles. A set of 5 unconstrained benchmark functions were used to ascertain if the algorithm is working according to specification or no. Bhuvaneswari et al. (2009) proposed an algorithm named HGAPSO which is a hybrid of genetic algorithm and particle swarm optimisation. The algorithm was used for alternator construction optimisation. Premalatha and Natarajan (2009) developed a DPSO algorithm that uses GA's mutation and crossover operators for document clustering. Abdel-Kader (2010) introduced a GAI-PSO algorithm that integrates the velocity and position update equations of PSO together with the concept of selection and crossover from GAs.

Hendtlass (2001) proposed the Swarm Differential Evolution Algorithm (SDEA) where PSO swarm acts as the population for Differential Evolution (DE) algorithm, and the DE is carried out over some generations. After the DE have performed its part in the optimisation, the resulting population is then optimized by PSO. Talbi and Batauche (2004) developed the DEPSO algorithm and used it to solve problem in the field of medical image processing. In Hao et al. (2007) introduced another variant of DEPSO where some probability distribution rules are used any of PSO or DE to produce the best solution. Omran et al. (2008) developed a Bare Bones Differential Evolution (BBDE) algorithm which used the idea of barebones PSO and self-adaptive DE approaches. They used their algorithm to solve image categorization problem. Jose et al. (2009) developed another variant of DEPSO algorithm that uses the differential modification systems of DE to update the velocities of particles in the swarm. Zhang et al. (2009) proposed the DE-PSO algorithm that uses three unconventional updating approaches. Liu et al. (2009) developed the PSO-DE algorithm that

combines DE with PSO and uses the DE to update the former best positions of PSO particles to make them escape local magnetizers thereby avoiding inertia in the population. Capanio et al. (2009) proposed a Superfit Memetic Differential Evolution (SFMDE) algorithm which is a hybrid of DE, PSO, Nelder Mead algorithm and Rosenbrock algorithm. The algorithm was used to solve some standard benchmark and engineering problems.

Xu and Gu (2009) developed the particle swarm optimisation with prior crossover differential evolution (PSOPDE). Pant et al. (2009) reported a DE-PSO algorithm that uses DE for the initial optimisation process and then moved on to the PSO segment if DE fails to satisfy the optimum conditions. Khamsawang et al. (2009) introduced another hybrid algorithm name PSO-DE that centres on standard PSO and DE. They used their algorithm to solve economic dispatch (ED) problem having constraints. Shelokar et al. (2007) developed PSO with Ant Colony Optimisation (PSACO) algorithm. The algorithm has two phases. The PSO is employed in the first phase and the result of the optimisation is feed into ACO for the second phase of the optimisation. Hendtlass and Randall (2001) integrated ACO into PSO. The best position is selected from the list of best positions obtained and recorded. Victoire and Jeyakumar (2004) proposed the hybrid of PSO and sequential quadratic programming (SQP). It was used to solve economic dispatch problem in Boggs and Tolle (1995). Grosan et al. (2005) developed an independent neighborhoods particle swarm optimisation (INPSO) algorithm that is made up of autonomous sub-swarms that allows the production of many points at the end of iteration.

Liu et al. (2007) developed a turbulent PSO (TPSO) in the effort to surmount the shortcomings of the traditional PSO. They later integrate TPSO with a fuzzy logic controller to make a Fuzzy Adaptive TPSO (FATPSO). Sha and Hsu (2006) proposed a novel hybrid algorithm that combine PSO with Tabu search (TS) and applied it to solve job shop problem (JSP). He and Wang (2007) proposed a hybrid algorithm that fuses PSO and Simulated

Annealing (SA) together. Mo et al. (2007) introduced Particle Swarm Assisted Incremental Evolution Strategy (PIES). This algorithm uses PSO for global optimisation while Evolutionary strategy is used for local optimisation. Fan and Zahara (2007) and Fan et al. (2004) proposed the NM-PSO algorithm by integrating PSO with Nelder Mead Simplex method. Their work was later extended by Zahara and Kao (2009) and used to solve constricted optimisation tasks. Guo et al. (2006) proposed an algorithm that hybridized PSO with gradient descent (GD) method and they used it for fault identification. Shen et al. (2007) introduced the HPSOTS algorithm which is a hybrid of PSO and Tabu search. Ge et al. (2008) developed a hybrid of PSO and Artificial Immune System (AIS). Song et al. (2008) developed hybrid particle swarm cooperative optimisation (HPSCO) algorithm merging simulated annealing algorithm and simplex method.

Kao et al. (2008) proposed an algorithm that combine NM-PSO algorithm developed by (Fan & Zahara, 2007) and (Fan, Liang, & Zahara, 2004), with K-means algorithm and used for data clustering. Murthy et al. (2009) proposed an algorithm that have the advantages of the parameter-free PSO (pf-PSO) and the extrapolated particle swarm optimisation like algorithm (ePSO). Kuo et al. (2009) proposed the HPSO algorithm that amalgamated a random-key(RK) encoding system, individual enhancement (IE) system, and particle swarm optimisation (PSO) and used to solve the flow-shop scheduling tasks. Chen et al. (2010) developed the PSO-EO algorithm by hybridizing of PSO with Extremal Optimisation (EO) as reported in (1999). Kaveh and Talatahari (2009a) and Kaveh and Talatahari (2009b) proposed a heuristic particle swarm ant colony optimisation (HPSACO) and a discrete heuristic particle swarm ant colony optimisation (DHPSACO). Wei et al. (2002) introduced the concept of entrenching swarm targets into Fast Evolutionary Programming (FEP) algorithm to make the swarm to perform better. Pant et al. (2008) presented an AMPSO algorithm which combines PSO and EP mutation operator employing Beta distribution.

The (VL-ALPSO) was proposed by Tang and Eberhard (2011) to make planning for change in the physical position of swarm robots for collective search of targets more effective. VL-ALPSO approach to swarm robotics is the amalgamation of augmented Lagrangian multipliers, velocity restrictions in addition to virtual detectors to guarantee the implementation of constraints, obstacle avoidance and mutual avoidance which are situations obtainable in swarm mobile robots in coordinated movements. Augmented Lagrangian Particle Swarm Optimisation (ALPSO) algorithm was presented by Sedlaczek and Eberhard (2005). They made use of some part of the original PSO technique and combines it with Augmented Lagrangian Multiplier.

PSO has been used in the area of robotics because of its ease of implementation, quick convergence and few control parameters that need to be adjusted. Hayes et al (2003) and Jatmiko et al (2007) used PSO to solve the odor localization problem using swarm of robots. Another researcher employed PSO for multi-robot searching though they failed to take into account the ability to handle growth in the number of robots which is an attribute of the typical PSO (Doctor et al, 2004). In 2006, a distributed particle swarm optimisation algorithm was developed by Hereford (2006), which they later improve upon in (Hereford & Siebold, 2008) and used for physical robots which can only make a circular movement in a definite angle. They however did not put into consideration the possibility of the robot encountering an obstacle in the search space (Tang & Eberhard, 2011). Pugh et al (2006) compared and contrast between physical robot and ideal particle in terms of their properties. They then broaden PSO to directly prototype many robots for investigating at a conceptual level the consequences of varying parameters of the swarm robots system (Pugh & Martinoli, 2007). A PSO algorithm was presented by Akat and Gazi in which the particle's neighbours or the arrangement of neighbours surrounding a particle is altered continuously as time goes on (Akat & Gazi, 2008).

A target search PSO algorithm was presented for swarm robotics that centres on an elaborate and systematic search of the search space by the robots. Simulations were done to control the movement of the robots asynchronously. Their work put into consideration the mechanical properties of the robot (Xue et al, 2009). However, there was no obstacle in their simulation environment, and the volume of the robot was not considered. Derr and Manic (2009) also demonstrated how the distributed PSO (dPSO) algorithm can be used in an environment that is very hazardous to direct the path of robots to find their goal(s). The algorithm after its development was tested using many robots to look for single target, and also more than one target. To increase the efficiency and effectiveness of the search many robots were introduced to do diverse search simultaneously in some environments that are characterized by strepent. It was established that the presence of a frequency of electromagnetic radiation in the range at which radio signals are transmitted can drastically influence the time it will take a robot to get to its desired goal. They however left out the mechanical properties of the robot and only concentrate on algorithms.

Doctor 2004 used PSO for path planning of unmanned vehicle that can converge very well. A soft and efficient path planning method for using Stochastic PSO was presented by Chen et al (2006). A Chaotic PSO was developed by Zhang, Wu and Wang (2013) for planning collision free path for robots. PSO was used to obtain the global best particle while local chaotic iterations are utilized to increase the accuracy of the solution. A path planning approach that used Chaotic PSO with mutation operator was proposed by Qin and Colleagues (2004). Also, Hao and Colleagues (2007) presented an approach that uses PSO and Polar coordinate system to avoid obstacles in in a non-stationary environment. PSO was used by Wang et al (2006) for controlling the direction and position of soccer robot. Karimu (2012) used dynamic hybrid PSO algorithm to solve motion planning problem of mobile robot. Qin et al (2004) proposed a path planning method based on PSO with mutation operator. The

approach uses the Dijkstra's algorithm to find the shortest path in the MAKLINK graph, and then used PSO to optimize the shortest path generated. The algorithm is suitable for stationary environment where the barriers are protrusive polygon in shape.

A multi-objective PSO for avoiding obstacles which are assumed to be circles in dynamic environment was proposed by Min, Zhu and Zheng (2005). PSO algorithm is used in getting the solution by adjusting the velocity of the robot. The limitations of this approach include the fact that path planning will be very difficult except the obstacle is a simple circle. And the approach may not produce the best result for concave barriers. Raja and Pugazhenti (2009) proposed a PSO planner for dynamic environment which can be used to obtain the optimum solution for the path that the mobile robot will navigate in the search space. Ellips and Davoud (2010) developed a multi-objective PSO-based algorithm for robot path planning. They used PSO for global path planning and the Probabilistic Roadmap Method (PRM) was used for obstacle avoidance. Yinghua Xue and Hongpeng Liu (2011) proposed a new variant of PSO that centres on the degree at which the obstacle changes. This approach is distributed thereby increasing the flexibility of the robot path planning in the search space. The strength of the algorithm is that its model is simple, convergences quickly, has automatic obstacle avoidance and can be used to generate the best path for the robot to navigate in various environments.

According to Basturk, and Karaboga, (2007), PSO is a stochastic population based algorithm that operates on the optimisation of a candidate solution (or particle) (Venayagomoorthy et al, 2008). The original PSO algorithm was introduced by Kennedy and Eberhart (1995). Their algorithm was centered on the social behavior demonstrated by a flock of bird, a school of fish, and herds of animals. The algorithm make use of particles that go through ongoing transformations by means of cooperation and competition among the particles from one generation to the other. PSO have been used to solve non-differentiable

(Guerra & Coelho, 2007), non-linear (Guerra & Coelho, 2007) and (Eberhart & Shi, 1998), and non-convex engineering problems (Rusman, 2013).

PSO is theoretically straightforward and does not require any sophisticated computation (Abraham, Konar & Das, 2008). PSO uses a small number of parameters, which have minima influence on the outcomes unlike any other optimisation algorithms. This property also applies to the initial generation of the algorithm. The randomness of the initial generation will not affect the output produced. Despite these advantages, PSO faces similar shortcomings as other optimisation algorithms. Specifically, PSO algorithm suffers from premature convergence, lack of capacity to provide solution to dynamic optimisation problems, the tendency of particles to be stuck in local minima and partial optimism (i.e., which worsens the control of its speed and direction). In swarm robotics, PSO particles moves within the search space to find optimal solution for the swarm by updating their velocity and position. According to (Xue et al, 2011) robots with actual velocities and physical positions that made up the swarm can be mapped to particles in PSO as they carry out their target search in the search space. The flowchart in figure 2.4 is the general steps of the PSO algorithm.

2.4 Variants of particle swarm optimisation algorithms in swarm robotics

In concordance with the development of a number of variant to reduce the disadvantage of general PSO, the application of PSO in swarm robotics has also spawned a number of dedicated PSO algorithms. These variants were developed to accommodate the different requirement of robotics tasks and characteristics. This is not limited to the extension from singular to multiple robots environment, but also includes the parallel operative nature of these swarm robots that work uniquely but at the same time in unison with the other robots. In this subsection, we describe some PSO algorithms that have been developed over the decades and are been used in the field of swarm robots.

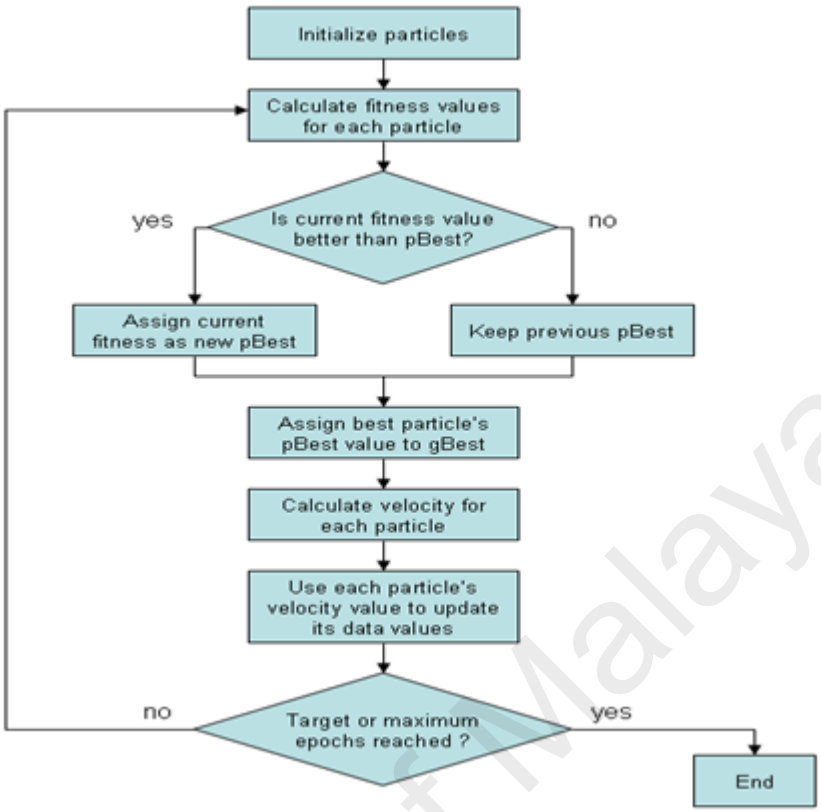


Figure 2.4: Procedure of the PSO algorithm

2.4.1 The Standard PSO

It was originally proposed by James Kennedy and Russell Eberhart in 1995 (Kennedy, Eberhart & Shi, 2004). The algorithm is made up of particles which have position and velocity. Each of the particles of a swarm epitomizes a possible solution in PSO. The particles explore the problem search space seeking for the best or at least a solution that is suitable. Each of the particles changes their movement according to their own accumulated knowledge of moving in the environment and that of their neighbours.

In PSO (X_i) represent the position of a particle, and (V_i) the velocity of the particle. The particle's number is i. Where (i = 1,…,N), and N is the number of particles in the swarm. The i^{th} particle is denoted as $X_I = (X_{i!}, X_{i2},..., X_{iN})$. The velocity is the degree at which the subsequent position is varying as regards the present position. $V_I = (V_{i!}, V_{i2},..., V_{iN})$ represent

the velocity for the particle i. As the algorithm begins, the position and velocity of the particles are given numerical values haphazardly. This is followed by using equations (1) and (2) to update the position and velocity of the particles after successive iterations are conducted throughout the search.

$$v_{i,m}^{(t+1)} = w * v_{i,m}^{(t)} + c_1 * rand_1() * (pbest_{i,m} - x_{i,m}^{(t)} + c_2 * rand_2() * (gbest_m - x_{i,m}^{(t)}) \quad (1)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (2)$$

Shi and Eberhart (2004) showed that PSO having different swarm population has practically alike but not identical performance. c_1 and c_2 are two positive constants representing the cognitive scaling and social scaling factors which according to Kennedy, Eberhart, and Shi (2001) are usually set to 2. The stochastic variable $rand_1()$ and $rand_2()$ has the distribution U (0, 1). These random variables are stand-alone functions that infuse momentum to the particles. The most ideal position located so far by the particle is denoted as $pbest_{i,m}$. The best position attained by the neighbouring particles is denoted as $gbest_m$.

There are two types of particles neighbourhood in PSO, and the type of neighbourhood is what determines the value of $gbest_m$. The two types of neighbourhood are:

1. *gBest* (Global neighbourhood) – Here, there is a full connection among the particles, and the exploration of swarm is controlled by the best particle in the swarm.
2. *lBest* (Local neighbourhood) – There is no full connection among the particles in the swarm, rather they are connected only to their neighbours.

Equation 2 is used in updating the position of the particles whereby the velocity is added together with the earlier position and a new search is started from its former position. Eberhart

and Shi (2000) in their work bounded $x_{i,m}^{(t+1)}$ to avoid a situation whereby particles are spending too much time in infeasible region. A problem dependent fitness function is used in evaluating the superiority of $x_{i,m}^{(t+1)}$. Assuming the present solution is superior to the fitness of $pbest_{i,m}$ or $gbest_{m}$ then the new position will replace $pbest_{i,m}$ or $gbest_{m}$ accordingly. Unless the condition for ending the search (either the iteration has reached its peak or we have gotten the desired solution) this updating process will continue. The optimal solution is the best particle found when the stopping criterion is satisfied (Aziz & Ibrahim, 2012). The flowchart for the original PSO for collective robot search is shown in Figure 2.4 above.

Where: i = particles identification counter form 1 to pop_size

X_i^d = ith particle's dth dimension's value

pop_size = Population size

gen_count = generation counter from 1 to max_gen

dimen = dimension

w = inertia weight (w_0 = 0.9, w_1= 0.4)

max_gen = maximum generations

$c_1 = c_2 = 2$

As opined by Shi and Eberhart (1998), to prevent commotion, the value $v_{i,m}^{(t+1)}$ is fixed at $\pm v_{max}$. The reason is that the value of v_{max} is going to be extremely large if the scope of search is too broad. Also, if v_{max} is very narrow, the extent of the search will be unreasonably reduced thereby forcing the particles to do local exploration. The inertia weight is represented as *w* (constriction factor) is the inertia parameter; this regulates algorithm's searching properties. Shi & Eberhart (1998) opined that it is better to commence the search using a larger inertia value (a more global search) that is automatically decreased to the end of the

optimisation (a more local search). Using inertia weight with smaller values mostly ensures fast convergence as little time is wasted on the exploration of the global space (Aziz & Ibrahim, 2012). The inclusion of *w* in the equation is to provide equilibrium between the global and local search capability of the particles. There are two techniques that have been presented for the choice of suitable values for inertia factor. The number one technique is called linear method, here the inertia weight decreases linearly after each iteration until the highest number of iteration or the highest number of inertia parameter is reached (Eberhart & Shi, 2001).

$$w_{i+1} = w_{max} - \frac{w_{max} - w_{min}}{i_{max}} i$$

The number two technique is called the dynamic method; here the value of the inertia reduces from the initial value to the final value fractionally by Δ_w.

$$w_{i+1} = \Delta_w w_i$$

Where the value of Δ_i varies from 1 to 0. Judging from the results of experiments that have been performed by Shi and Eberhart (2004), the performance of the dynamic method in term of convergence is superior to the one of linear method because it dynamically reduces the value of the inertia weight from maximum to minimum.

2.4.2 Synchronous PSO (SPSO)

Particles in the conventional PSO perform synchronous updates, i.e., the best particle in each neighborhood is located and then used by the other particles to update their positions. The entire information of the neighbours is possessed by all the particles as reported in the work of Juan, Mengjie, and Winston (2011). $pbest_{i,m}$ or $gbest_m$ update are done after all particles' fitness have been evaluated. This type of approach has the advantage of quick convergence and good result. Carlisle and Dozier (2001) however observed that the

synchronous update is costly as the first particle evaluated will be redundant for some time since it has to wait for other particles to be evaluated before it can progress to a another position and continue exploring the search domain.

2.4.3 Asynchronous PSO (APSO)

In the Asynchronous PSO (APSO), $pbest_{i,m}$ and $gbest_{m}$ of a particle, its velocity and position are updated immediately after computing their fitness and, as a consequence, they update it having incomplete or imperfect information about the neighbourhood. This result into varieties in the swarm since some of the information is from the previous iteration while some is from the current iteration. In the work of Luo and Zhang (2006), they used the bench-mark functions of Rosenbrock (unimodal) and Griewank (multimodal) to do a performance comparison of SPSO and APSO on the Rosenbrock (unimodal) and Griewank (multimodal) bench-mark functions. They found out that APSO performs better and has a faster convergence than SPSO. Perez and Basterrechea (2005) opined from the results of their experiments that APSO is able to find solutions faster and with a similar accuracy as SPSO. They concluded that APSO provides the best accuracy at the expense of computational time.

2.4.4 Extended Particle Swarm Optimisation (EPSO)

Extended Particle Swarm Optimisation (EPSO) was proposed by Pugh and Martinoli (2006, 2007). In the work of Jun-jie and Zhang- hong (2005), the EPSO algorithm provides a platform whereby each of the robots are directed through unknown environment to their goal by their individual intuition knowledge (cognitive) and accumulated knowledge (social experience) of their companionship with other robots in the swarm. EPSO utilizes the existing advantages of *gBest* and *lBest* in previous versions of PSO. In the EPSO algorithm, the robots are not connected in a multi-step nature and there is no restriction on the movement

of the robots in the swarm thereby guaranteeing greater level of communication among the robots. The Braitenberg obstacle avoidance algorithm (Braitenberg, 1984) was incorporated into the main equation and used to achieve obstacle avoidance mechanism in the EPSO algorithm. The robot will update its inner velocity and keep heading towards a different direction if it encounters an obstacle by implementing a step of the algorithm. The advantage of this technique is that it allows the collision prevention procedures to be treated separately from some of the other eminent characteristics of robots. However, this approach is not likely to be practicable if there is the need to analyze the steadiness of the algorithm in view of the effect the barriers have on the robots (Couceiro et al, 2012c). Moreover, it will be difficult to define a technique that can fit into all situations whereby it will be able to use appropriate information to modify all the parameters for the algorithm (Couceiro et al, 2012a).

2.4.5 Group Decision Making Extended Particle Swarm Optimisation (GDMEPSO)

Xue Songdong et. al. (2012) proposed the Group Decision Making Extended Particle Swarm Optimisation (GDMEPSO) algorithm, which is a fully distributive algorithm and was proved to be effective even when the size of the swarm is too large. They modified the theory behind EPSO by exchanging the social experience with the approximation value of location of the desired goal. The swarm robots were mapped to WSN (Wireless Sensor Network). To enhance the efficacy of the search, the RSSI (received signal strength indication) wireless sensor network technique was considered as a merger of aggregate selection of course of action among several alternatives.

2.4.6 Multi-Robot, Multi-Target PSO

Multi-Robot, Multi-Target PSO algorithm was proposed by Kurt Derr and Milos Manic (2009). They developed a distributed PSO that utilizes multiple small mobile robots to search an unfamiliar terrain with the aim of locating the target(s). The PSO algorithm made

use of a new adaptive RSS (received signal strength) as a very important element that direct the movement of robots toward their goal(s) in a highly risky environment. Their experimental results show that electromagnetic wave frequency between audio and infrared can have a dramatic effect on the time it takes the robot to reach its target.

2.4.7 Physically embedded PSO (pePSO)

The Physically embedded PSO (pePSO) algorithm was proposed by Hereford, and Siebold, (2008). They employed two search strategies. First, the swarm robot moves throughout the search space and take measurements as they move towards their targets. Second, trophallactic, (which is the exchange of vomited partially digested food that occurs between adults and larvae in colonies of social insects) was developed into an algorithm and utilized for the search. The second search algorithm is advantageous in that no robot-to-robot communication is needed; communication radius, protocol, or bandwidths are all unnecessary. Also, it is not necessary for the robots to have knowledge their position explicitly. The pePSO does not make use of any main agent to direct the movements and behaviours of the robots in the swarm. Unlike what we have in the standard PSO, the movement of particles is confined within a regulated space to circumvent the no direct movement obtainable in the traditional PSO. The pePSO algorithm presumed that all the robots in the swarm are in harmony and it is after the transferring of all relevant information among the robots that the calculation of the robot's new position is done. Moreover, robots can only disclose information about their individual solution if their personal solution is the ideal solution for the entire swarm. The advantage of this approach is that it drastically decrease the volume of information interchange among robots. The approach can however lead to redundancy on the part of the robots as they would have to remain idle for some time

after successfully completing an iteration to process all appropriate information (Couceiro et al, 2013).

2.4.8 Distributed PSO (dPSO)

The Distributed PSO (dPSO) was proposed by Hereford, J. M. (2006) as an algorithm that can adequately handle a swarm made up of a considerable number of miniature robots. This variant of PSO algorithm is very efficient in locating the search goal. This algorithm makes provision for computation of the individual robot's current position. The exchange of information and signals among the robots in the swarm will be greatly reduced. Also, there is no need of a representative robot to organize the robots in the swarm into a harmonious movement towards their target.

2.4.9 Augmented Lagrangian PSO with Velocity Limits (VL-ALPSO)

Kai Sedlaczek and Peter Eberhard (2006) presented the Augmented Lagrangian Particle Swarm Optimisation (ALPSO) algorithm. They made use of some parts of the original PSO technique and combine them with Augmented Lagrangian Multiplier. The Augmented Lagrangian Multiplier is a comprehensive non-stationary penalty function method, which will give suitable result. The authors concluded from their experiments that ALPSO have quick convergence properties and it is a powerful tool for solving problems in real life applications that have few solutions. The drawbacks of this algorithm are the performance is poor when a fully connected topology (global best) is used, and conflicting situations based on information from many neighbors. This approach however allows informed individuals to find better solutions, as it is more likely in the neighborhood to have a particle with a high quality. Tang and Eberhard (2011) proposed the Augmented Lagrangian PSO with Velocity Limits (VL-ALPSO). The algorithm was proposed to handle

changes in the physical position of swarm robots for collective search of targets to be more effective. VL-ALPSO approach to swarm robotics is through the amalgamation of augmented Lagrangian multipliers, velocity restrictions in addition to virtual detectors to guarantee the implementation of constraints, obstacle avoidance and mutual avoidance, which are situations obtainable in, swarm mobile robots in coordinated movements. The algorithm is decentralized and the mechanical properties of the robots are taken into consideration.

2.4.10 Detection and responding PSO (DR PSO)

Detection and responding PSO (DR PSO) algorithm was presented by Jatmiko, Sekiyama and Fukuda in 2006. According to Andries P. Engelbrecht (2005), the complications of the dynamic environment cannot be solved by the original PSO. There is therefore the need to modify or improve on the original PSO so that it can solve problems that are dynamic in nature. Eberhart and Shi (2001), and Hu and Eberhart (2002) suggested that this could only be achieved through the integration of a system that can discover modification and react to it favorably. The global best information *gBest* is monitored by the alteration detection function. After some certain number of iterations, if *gBest* is unchanged, it possibly means that another optimum solution exists. Some productive plan for reacting positively to diversities of environmental fluctuations must be employed when environmental changes are detected. DR-PSO's inability to survive extreme alterations is the main drawback of this technique.

2.4.11 Charged PSO (CPSO)

Jatmiko, Sekiyama and Fukuda presented the Charged PSO (CPSO) algorithm in 2006 alongside DR-PSO to provide solution to the circulation of odor with time in a dynamic

environment. CPSO algorithm adopted the idea of Coulomb's inverse square law that described the electrostatic interaction between electrically charged particles. The charged particle here is represented as charged robot (which is used in CPSO). A non-attraction force is used on the charged robot while such force is not used on the neural robot. There is variety in the positional allocation of robots to avoid being snared in a local maximum. With this, the problem of inability to cope with extreme changes that characterized the DR-PSO is deemed solved.

2.4.12 Augmented Lagrangian Particle Swarm Optimisation (ALPSO)

This algorithm was presented by Sedlaczek and Eberhard (2005). They made use of some part of the original PSO technique and combines it with Augmented Lagrangian Multiplier. The Augmented Lagrangian Multiplier is a comprehensive non-stationary penalty function method which will give suitable result. The authors concluded from their experiments that ALPSO have quick convergence properties and it is a powerful tool for solving problems in real life applications that have few solutions.

2.4.13 Fully Informed Particle Swarm Optimisation Algorithm (FIPS)

Mendes et al. (2004) introduced the Fully Informed Particle Swarm Optimisation (FIPS) algorithm. It is a variant of PSO algorithms that exploit the velocity update approach of the best neighborhood. This algorithm readily reacts to alterations in the configuration of the population. All neighbors of a particle are carefully considered before updating their velocity. This is unlike some variants of PSO that only update the velocity of the best neighbor. FIPS performs better when the topologies have a lower degree such as the ring lattice topology (local best) or topologies where the particles have very few neighbors (not more than three).

2.4.14 Robotic Darwinian PSO (RDPSO)

Couceiro, Rocha, and Ferreira (2011a & 2011b) as an extension of the Darwinian PSO (DPSO) presented the Robotic Darwinian PSO (RDPSO) algorithm. This algorithm like the standard PSO, is made up of a swarm of robots that moves as a group in the search space to locate the ideal solution. Each robot have a position, the direction they are going, and their performance. The RDPSO permits the swarm to be divided into various dynamic groups of sub swarm. This supports a distributed method rather than the centralized method obtainable in some PSO algorithms where the network is likely to have covered the whole swarm of robots. The advantage of this is that with the swarm divided into smaller groups of swarms, there is a decrease in the robots that are needed to get the ideal solution and the volume of exchanges information among robots also decrease thus reducing the overhead cost.

Moreover, that means that dividing the robot swarm into mutually exclusive categories is an added advantage for RDPSO since the volume of information transfer among the robots will decrease to the barest minimum. The algorithm does not need any central agents to coordinate robots' movements or actions. The RDPSO is highly scalable thereby allowing the addition of a huge number of robots to the swarm. The weaknesses of the algorithm includes the lack of adaptability to contextual information, and the changing over time of the sub-optimal solutions which according to J. Suarez, R. Murphy (2011) can be overcome by sweeping the whole scenario with robots. Couceiro et al (2012a, 2012b, & 2012c) in some of their most recent research demonstrated that the RDPSO can solve some problems related to swarm robotics such as obstacle avoidance, dynamic nature of the robots in the search space, finding ideal solutions, and ability to handle some of the communication restraints (Couceiro et al, 2013).

Table 2.1: A summary of variants PSO algorithms implementation specifically for swarm robotics.

S/No	Technique	Author(s) and Year	Strength(s)	Weakness(es)	Description
1	Original PSO	Kennedy and Eberhert (1995)	Fast convergence, simple, effective, and easy to implement without a complex calculation.	Premature convergence, inability to solve dynamic optimisation problems, and cannot be scaled to accommodate large number of robots.	Uses particles that have position and velocity rather than genetic operators. Inertia weight *w* is in the range of 0.2 to 0.4, while c1 and c2 equal to 2.
2	Synchronous PSO (SPSO)	Kennedy and Eberhert (1995)	Quick convergence, it produces good result, and it can be adapted to solve a given problem.	It is costly as the first particle evaluated will be redundant for some time since it has to wait for other particles before its position can be updated. It also has poor parallel competence.	Inertia weight *w* is in the range of 0.4 to 0.9, while c1 and c2 equal to 2.

3	Asynchronous PSO (APSO)	Koh, Fregly, George, and Haftka (2005)	Fully utilize the processor as there is no idle processor.	Provides the best accuracy at the expense of computational time.	Inertia weight *w* is dynamic, while c1 and c2 = 2
4	Extended PSO (EPSO)	Jun-jie and Zhang- hong (2005)	Utilize the existing advantages of gBest and lBest version of PSO. It has less computational costs. It can also be easily mapped to robot or swarm of robots.	It cannot be used to solve complex dynamic optimisation problems. Also, it cannot handle large number of robots in the swarm.	Makes use of intuition knowledge to direct the movement of robots to their search goal.
5	Group Decision Making Extended PSO (GDMEPSO)	Xue, Zan, Zeng, Xue and Jing (2012)	Is very effective, can successfully handle large number of robots in the swarm. The guiding effect of GDMEPSO is strong than that of EPSO	As the swarm size becomes very big, the efficiency of GDMEPSO becomes lower than that of EPSO. Energy consumption increases as swarm size increases.	It is a fully distributive algorithm where communication of swarm robots is taken as wireless sensor network.
6	Multi-Robot Multi-Target PSO	Kurt Derr and Milos Manic (2009)	It can search for multiple targets in a noisy environment using many robots in the swarm.	Signal degeneration can adversely affect robots' navigation.	It uses the combination of decentralised PSO algorithm and new adaptive RSS to enable

			It can be used for hazardous target search. Overshooting of target is greatly prevented.		robots to locate their target(s).
7	Physically embedded PSO (pePSO)	Hereford and Siebold (2008)	Has the ability to locate target in complicated environment. The need for robot to robot communication is completely eradicated.	Search time can be unpleasantly long when there are obstacles in the search space. Weak communication signal can unfavourably affect search time.	Trophallactic behaviou r in social insects was developed into an algorithm. The algorithm allows robots to take measurement in the search space from the location where they are to the target.
8	Distributed PSO (dPSO)	Hereford (2006)	It is computationally simple. It can efficiently handle swarm that is made up of large number of robots.	The time taken to locate the target is much because the mobility on the path of the robots are limited	It is a distributive algorithm and there is no need of any representative robot.

			It has the ability to locate targets efficiently. Inter-robot communication is minimized.		
9	Augmented Lagrangian PSO with Velocity Limits (VL-ALPSO)	Tang and Eberhard (2011)	It is simple and reliable. It has resilient convergence. It is scalable to a very huge number of robots. It can handle diversity in swarm thereby solving the problem of premature convergence. It can be modified to evade barriers that are dynamic in nature in the search space.	There is non-monotonous decrease in the value of the objective function. Requires making a lot of adjustment to get the best performance for it.	Uses a decentralised algorithm. The technique employed the combination of augmented Lagrangian multiplier with velocity restrictions and virtual detectors.
10	Detecting and Responding PSO (DR PSO)	Jatmiko, Sekiyama and Fukuda (2006)	It can solve the complexity problem in the dynamic environment.	Cannot survive extreme alterations in dynamic environment.	It uses change detection function to monitor gBest.
11	Charged PSO (CPSO)	Sekiyama and Fukuda (2006)	It solved the problem of inability to cope	The instabilities in the charged swarm	It adopted the idea of

			with extreme changes that characterized DR PSO.	size can be very large. Introduction of charged particles makes it to be wasteful. The algorithm is not scalable.	Coulomb's law. A charged robot is introduced with a repulsive force and a neutral robot with non -repulsive force.
12	Augmented Lagrangian PSO (ALPSO)	Sedlaczek and Eberhard (2005)	Quick convergence. Powerful tool for solving constrained problems.	There is problem of incomplete information since individual robot only have information on the value of self-best and swarm best.	Makes uses of classical PSO and Augmented Lagrangian multiplier.
13	Fully Informed PSO (FPSO)	R. Mendes, J. Kennedy, and J. Neves. (2004)	Performance is high with lower degree topology. It is simpler.	High computational cost is associated with the algorithm. Performs poorly when global best is used. Information from neighbourhood can lead to conflict. Performance can be adversely affected by	It exploits the velocity update of its neighbourhood after careful consideration.

				changes in topology of the population.	
14	Robotic Darwinian PSO (RDPSO)	Couceiro, Rocha and Ferreira (2011)	It is highly scalable. The need for central coordinating robot is eliminated. The volume of information transfer from one robot to another is reduced to the barest minimum.	Lack of adaptability to contextual information. Frequent changes of sub optimal solutions with time.	It uses sociobiological technique to improve the ability of canonical PSO to avoid been trapped in local optimal.

2.5 Interior-Point Methods (IPMs)

The notion of the "interior point" method was derived from the linear programming annotation. The Interior Point Methods go through the feasible search space to arrive at the most favourable solution. This is contrary to what we have in the simplex method which go along a succession of contiguous extreme points to the best solution. Interior Point Methods are normally grouped into three principal classes: Affine-scaling methods, primal-dual methods, and Projective methods. From the aforementioned types of IPMs, the one that is most popular for achieving maximum productivity is the primal-dual (as well as primal-dual algorithms that integrate predictor - corrector) algorithms. Quintana and Torres (1997) reported that the most important stages in all Interior Point Methods are: Converting the inequality constrained optimisation problem to equality constrained one, use the logarithmic barrier functions to develop the Lagrange function, decide on the first-order most favourable conditions, and utilize the Newton's method to the group of equations coming from the first-order most favourable conditions.

Since the 1960s, different variants of the Interior Point Methods have been developed for solving nonlinear and linear programming problems. For more than 25 years now, researchers and academicians have contributed immensely to the development of the Interior Point Methods thereby making it a practicable set of algorithms that are suitable for solving different kinds of optimisation problems that are conical in shape (Quintana and Torres, 1997). According to (Dikin, 1967), the lack of convergence in the data of the problem constitutes the early shortcoming in the use of Interior Point Methods for Linear Programming. Many of the interior-point methods employ a revised Newton method to decide the direction of search for each iteration. The set of equations relating to the revised Newton system can then be summarized to a scheme of equations whose matrix AD^2A^T is positive but habitually not in peak condition (Jorge & Stephen, 2006). The logic backing the development of these methods originated from the truth that the optimal conditions for linear programming known as the Karush-Kuhn-Tucker (KKT) conditions, is expressed as

$$\{z : F(x) = 0, G(x) \geq 0\},$$

where $F(\cdot)$ and $G(\cdot)$ are maps and z is a vector. Interior point methods are iterative algorithms generating a sequence of points $\{s^j\}$ lying in the interior of the set $\{z: G(x) \geq 0\}$ (hence the name), and then come together at a point s^* fulfilling the KKT conditions.

It have been shown beyond reasonable doubt in the work of (Jorge & Stephen, 2006) that Interior-Point Methods (IPMs) also known as barrier methods are powerful tools for providing solution to nonlinear optimisation problems. The IPMs and SQP methods have constrains that are functional at the present stage are now believed to be the best algorithms for providing solution to large-scale nonlinear optimisation problems. Though they are very efficient, but they are still plagued with several challenges such as how to deal with non-convexity, the way to update the barrier constraint notwithstanding the presence of

nonlinearities. Also, is the necessity of making sure the algorithm advance in the direction of the solution. The continuation methods used by IPMs include the linear and quadratic programming which can be classified as an extended part of interior-point methods that impose convergence using line searches and engage direct linear algebra for steps computation. IPMs also make use of the barrier methods which use a trust region constraint to keep stability while the quadratic model is used to define the step (Jorge & Stephen, 2006).

Ben-Tal and Nemirovski (2001) pointed out that one of the strength of the IPMs is their polynomial convergence properties. Karmarkar (1984) in his work proved that an interior-point method for linear programming have verifiable polynomial convergence qualities. $O(n^{2.5}\log \epsilon^{-1})$ arithmetic operations is needed by the algorithm to attain an ϵ-solution to linear programming, where the size of the decision vector is n. Also IPMs have the ability to solve problems that are classified as complex. Particularly, IPMs are been used in solving problems in conic programming that are of the form

$$\min \{c^T x : Ax - b \in K\},$$

where K is a cone that is closed, convex, and have non-empty interior (Renegar, 2001). The most popular categories of conic problems for which IPMs have been developed are

1. Linear Programming (LP): $K = R_+^n = \{x \in R^n: x_i \geq 0 \text{ for all } i = 1..., n\}$,

2. Second-Order Cone Programming (SOCP): $K = K_1 \times \cdots \times K_p$, where each

$$K_i = \{x \in R^{ni}: x_{ni} \geq \sqrt{x_i^2 + \cdots + x_{ni-1}^2}\}, \text{ and}$$

3. Semi-Definite Programming (SDP): $K = \{X \in R^{n \times n}: X = X^T, y^T X y \geq 0 \text{ for all } y \in R^n\}$.

Ben-Tal and Nemirovski (2001) concluded that LP $\subset$ SOCP $\subset$ SDP when some simple changes are made to the variables.

Interior-point methods (IPMs) when contrasted with active set and cutting plane methods flick the amount of work; thereby making the number of iteration to be lower both theoretically and practically than the one we have in active set methods. When the interior-point methods is used for solving problems in linear programming, for a hard-and-fast precision, a $O(\sqrt{n})$ iterations is enough hypothetically, but in practice it is deemed that only O(log n) iterations are needed. One of the shortcomings of Interior-Point methods is the excessive work per iteration that is commonly required. For a linear programming problem, we can have the following:

$$\min c^T x \qquad\qquad \max b^T y$$

$$\text{s.t. } Ax = b, \qquad \text{and} \qquad \text{s.t. } A^T y \leq c.$$

$$x \geq 0,$$

Despite the fact that there are many types of interior-point-algorithms that can be used to solve linear programming problems; unless the right linear algebra methods are employed, the principal work per iteration is the development and result of a system of "standard equations", which can be written as:

$$ADA^T u = f,$$

where A = the constraint matrix of the LP, and

D = some diagonal matrix.

The attached costs required to develop the normal matrix is $O(nm^2)$ floating-point computations and that of solving the resulting system, using Cholesky factorization ($A = LL^T$) and back substitution is $O(m^3)$ when A is complex.

2.6 The Barrier Methods

The 1960s witnessed the development of the barrier methods though its popularity has dwindled over time. The efficiency of the interior-point methods for solving linear programming problems has inspired new attention from researchers and Mathematicians in using barrier methods for solving nonlinear problems. The have led to the development of more innovative techniques and software for nonlinear programming at the end of the 1990s. Boyd and Vandenberghe (2004) have opined that Interior Points Methods make use of barrier function in F to convert constrained optimisation problem into a series of unconstrained problems. This makes the gradient of the augmented criterion unbounded at the boundary of the feasible domain so that its minimizers fulfill the constraints. Results of experiments conducted have shown that Interior-Point Methods are faster in solving large scale problems with thousands of free variables and constraints compared to active-set Sequential Quadratic Programming methods. They are likely not too robust as the problem they are to solve becomes larger. The terms "interior-point methods" and "barrier methods" are now used interchangeably.

$$\min_{x,s} f(x) \qquad (2.1a)$$

$$\text{subject to } C_E(\mathrm{x}) = 0 \qquad (2.1b)$$

$$C_I(\mathrm{x}) - \mathrm{s} = 0 \qquad (2.1c)$$

$$\mathrm{s} \geq 0. \qquad (2.1d)$$

$C_{\mathrm{I}}(x)$ is a vector that is created from the scalar functions $Ci\ (x)$, $i \in I$. The same thing applies to $C_{\mathrm{E}}(x)$. From the above equation, the insertion of a vector s of idle variables into the inequalities $C_{\mathrm{I}}(x) \geq 0$ changed it into equalities. The variable l was used to represent the number of equality constraints which is the dimension of the vector C_{E}, while m is the

dimension of C_I and was used to represent the number of inequality constraints. Interior-point methods can also be categorised as the barrier methods or as continuation methods. The Karush-Kuhn-Tucker (KKT) conditions for the nonlinear program (2.1) can be written as

$$\nabla f(x) - A_E{}^T(x)\, y - A_I\, T(x)\, z = 0, \qquad (2.2a)$$

$$Sz - \mu e = 0, \qquad (2.2b)$$

$$C_E(x) = 0, \qquad (2.2c)$$

$$C_I(x) - s = 0, \qquad (2.2d)$$

with $\mu = 0$, together with

$$s \geq 0,\ z \geq 0 \qquad (2.3)$$

From the above equation, there are two Jacobian matrices $A_E(x)$ and $A_I(x)$ having C_E and C_I as functions respectively. The Lagrangian multipliers are y and z. The diagonal matrices S and Z are having diagonal entries s and z, where $e = (1,\ 1,\ \ldots,\ 1)^T$. Looking closely at the equation (2.2b), when the value of $\mu = 0$, the boundary limits specified in (2.3) help in initiating an ordered sequence of defining the best active set of the problem. The variables s and z are forced to have positive values by making μ to be always positive. The continuation approach is also called homotopy where one path can be continuously deformed into the other leaving the endpoints fixed and remaining within its defined region. The idea of homotopy is what gave birth to the description of primal-dual trend just as the barrier viewpoint is central in the scheme of having iterations that have global convergence.

This approach nearly solves the *disconcerted KKT conditions* (2.2) for series of positive variables $\{\mu k\}$ that tends to zero, while $s,\ z > 0$. This is to ensure that we find a spot where the KKT conditions for the nonlinear program (2.1) is met. The probability of the

iterate converging to a minimizer is very high as it becomes necessary for the iterate to reduce a merit function. This approach of two continuous functions from one topological space is locally acceptable. In a situation where ($x*$, $s*$, $y*$, $z*$) is a neighbourhood of solution that suits the self-sufficient linear constraint condition, the exact supplementary condition, and the second-order satisfactory conditions. There is a small positive value for μ in the system which has a local solution that is distinctive. These solutions are represented by ($x(\mu)$, $s(\mu)$, $y(\mu)$, $z(\mu)$). The primal-dual central path is the name of the path depicted by these positions, and it converges to ($x*$, $s*$, $y*$, $z*$) as $\mu \rightarrow 0$.

The second derivation of interior-point methods associates with (2.1) the barrier problem

$$\min_{x,s} f(x) - \mu \sum_{i=1}^{m} \log s_I \qquad (2.4a)$$

Subject to $C_E(x) = 0$, (2.4b)

$C_E(x) - s = 0$, (2.4c)

$\nabla f(x) - A_E{}^T(x)\, y - A_I\, T(x)\, z = 0$, (2.5a)

$\mu S^{-1} e + z = 0$, (2.5b)

$C_E(x) = 0$, (2.5c)

$C_I(x) - s = 0$, (2.5d)

The only difference between the above equations (2.5) and the one of (2.2) in only the second equation because it no longer linear as $s \rightarrow 0$ nears the solution. By multiplying the equation (2.5b) by S, it can be changed into a quadratic equation using Newton's method. However, since the diagonal components of S are non-negative, this multiplication does not

have any visible effect on the solution. After equation (2.5b) has been changed to a quadratic equation, the KKT specifications for the barrier problem correspond with the disconcerted KKT structure of equation (2.2).

According to Fiacco and Mccormick (1990), the "interior point" terminology originated from the reality that the barrier methods used in those days are without any slacks and assumed that the initial point x_0 is feasible with respect to the inequality constraints ci $(x) \geq 0$, $i \in$ I.

The barrier function

$$f(x) - \mu \sum_{i \in I} \log c_i (x)$$

is applied by these techniques to inhibit a situation in which the iterates will exit the feasible locality that have been identified by the lack of equalities. A good number of the state-of-the-art interior-point methods are impracticable. The implication of this is that they can begin from any original point x_0 and continue being interior for as long as the constraints $s \geq 0$, $z \geq 0$ remain. Certain modifications can however be made in the design so that when a feasible iterate is produced, all the succeeding iterates will continue to be feasible. However, they can be designed so that once they generate a feasible iterate; all subsequent iterates remain feasible vis-à-vis the lack of equalities.

2.7 Basic Interior-Point Algorithm

By using Newton's method for the nonlinear system (2.2) having the variables *x, s, y, z*, the result is the equation below:

$$\begin{bmatrix} \nabla^2_{xx} L & 0 & -A_E^T(x) & -A_E^T(x) \\ 0 & Z & 0 & S \\ A_E(x) & 0 & 0 & 0 \\ A_I(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} P_x \\ P_s \\ P_y \\ P_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A_E^T(x)y - A_I^T(x)z \\ Sz - \mu e \\ C_E(x) \\ C_I(x) - S \end{bmatrix} \quad (2.6)$$

where L represents the Lagrangian for (2.1a)–(2.1c):

$$L(x, s, y, z) = f(x) - y^T c_E(x) - z^T (c_I(x) - s). \quad (2.7)$$

Unlike the primal system that was reviewed in Section 2.3, the above system (2.6) is called the primal-dual system. After determining the step $p = (p_x , p_s, p_y , p_z)$, the new iterate (x^+, s^+, y^+, z^+) is calculated as:

$$x^+ = x + \alpha_s^{max} p_x, \qquad s^+ = s + \alpha_s^{max} ps, \quad (2.8a)$$

$$y^+ = y + \alpha_z^{max} p_y, \qquad z^+ = z + \alpha_s^{max} pz, \quad (2.8b)$$

where

$$\alpha_s^{max} = \max\{ \alpha \in (0,1] : s + \alpha ps \geq (1 - r)s\}, \quad (2.9a)$$

$$\alpha_z^{max} = \max\{ \alpha \in (0,1] : z + \alpha pz \geq (1 - r)z\}, \quad (2.9b)$$

while $\tau \in (0, 1)$, 0.995 is an ideal value for τ. The criteria stated in (2.9) are referred to as the fraction to the boundary rule. It was introduced to stop the variables s and z from prematurely getting to their lower bound which is 0. The fundamental principles of the state of the art interior point methods were based on this iteration. It has however gone through several alterations to make the interior point method to cope with non-convexities and nonlinearities. Another very important factor is the process of selecting the arrangement of barrier

parameters $\{\mu k\}$. Fiacco and McCormick (1998) in their study revealed that unless some level of accuracy is achieved for the KKT conditions specified in (2.2), the barrier parameter μ remains constant for the sequence of iterations. Another efficient way of doing this is to make the barrier parameter up to date for every iteration.

What we have in (2.2) is a primal dual matrix that continues to have a determinant that is not zero when the convergent solution from the iteration fulfills the conditions of the second-order sufficiency and strict complementarity. The nonsingular primal-dual matrix in (2.6) continues to be so and the iteration will converge to a solution that meets the second-order satisfactoriness conditions and meticulous off setting of mutual lack. Particularly, let us assume that x* is a solution whereby meticulous off setting of mutual lack is in place, then for every index i either s_i or z_i continues to be restricted from getting to zero as iterates move toward x*. This will ensure that the second block row of the primal-dual matrix (2.6) has full row rank. A superior rate of convergence can easily be created because the interior-point method does not produce ill conditioning or singularity.

The above discussion can be summed up with a description of the implementation of the basic interior-point method. The subsequent error function, which is founded on the disconcerted KKT system (2.2) was used:

$$E(x, s, y, z;\mu) = \max \{\|\nabla f(x) - A_{\mathrm{E}}(x)^T y - A_{\mathrm{I}}(x)^T z\|, \|S_z - \mu e\|, \|c_{\mathrm{E}}(x)\|, \|c_{\mathrm{I}}(x) - s\|\}, \qquad (2.10)$$

representing certain vector pattern $\|.\|$.

Algorithm 2.1 (Elementary Interior-Point Algorithm).

Select x_0 and $s_0 > 0$, and calculate first values for the multipliers y_0 and $z_0 > 0$.

Pick the original barrier parameter $\mu_0 > 0$ and parameters $\sigma, \tau \in (0, 1)$.

Let $k \leftarrow 0$.

while

condition for ending for the nonlinear program (2.1) is yet to be satisfied

repeat

while

$E(x_k , s_k , y_k , z_k ;\mu_k) \geq \mu_k$

repeat

Provide solution to (2.6) to find the search direction $p = (p_x, p_s, p_y, p_z)$;

$Calculate \propto_s^{max}, \propto_z^{max}$ applying (2.9);

$Calculate$ $(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1})$ applying (2.10);

Let $\mu_{k+1} \leftarrow \mu_k$ and $k \leftarrow k + 1$;

end

Select $\mu_k \in (0, \sigma\mu_k)$;

End

When we eliminate the prerequisite that the KKT conditions must fulfill for each μ_k in the inner "while - repeat" loop, we can without doubt get the up-to-date of the barrier parameter μ_k of each iteration from the Algorithm 2.1. This can be achieved by updating μ_k in the second to last line when a dynamic rule is employed.

The quest to make the interior point algorithm to be able to solve myriad problems which include the non-convex nonlinear problems have made researchers to make some amendments and augmentations on the Algorithm 2.1. Many a times, the primal-dual system (2.6) is changed to the symmetric model

$$\begin{bmatrix} \nabla^2_{xx} L & 0 & -A_E^T(x) & -A_E^T(x) \\ 0 & \sum & 0 & -I \\ A_E(x) & 0 & 0 & 0 \\ A_I(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} P_x \\ P_s \\ -P_y \\ -P_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A_E^T(x)y - A_I^T(x)z \\ Z - \mu S^{-1}e \\ C_E(x) \\ C_I(x) - S \end{bmatrix} \quad (2.11)$$

Where

$$\sum = S^{-1}Z \quad (2.12)$$

The workload involved in the calculation of each iteration is drastically reduced because a symmetric linear equation solver is used.

2.8 Primal and Primal-Dual System

2.8.1 Primal methods

Barrier methods actually acted in the area of primitive variables *x* before the launch of the primal-dual interior point methods. The purpose of the primal methods was to provide solution to nonlinear programming problems through unrestrained reduction utilized in a parametric sequence of functions. Inequality constrained problems can simply be explained using Primal barrier methods as follows:

$$^{min}_{x} f(x) \text{ subject to } c(x) \geq 0 \quad (2.13)$$

While the barrier-log function is described by

$$P(x; \mu) = f(x) - \mu \sum_{i \in I} log c_i(x) \quad (2.14)$$

as $\mu > 0$. It can be proved that $x(\mu)$ which minimizes $P(x; \mu)$, move towards a solution of (2.13) as $\mu \downarrow 0$, given some conditions. The path followed by the projectile C_p is stated as

$$C_p \stackrel{def}{=} \{x(\mu) | \mu > 0\} \tag{2.15}$$

this is usually described as the pivotal path of the primal.

We can search for $x(\mu)$ which minimizes $P(x; \mu)$ seeing that it is positioned in the firmly viable set $\{x \mid c(x) > 0\}$ where there is no operational restraints. We can use any of the unconstrained minimization algorithms to search for $x(\mu)$. To avoid a situation in which some of the steps will leave the feasible area or are too near to the restriction borders, there will be need to make some alteration in the algorithm. The approximate value of the Lagrange multipliers can be obtained by finding the differential value of P to get

$$\nabla_x p(x; \mu) = \nabla f(x) - \sum_{i \in I} \frac{\mu}{c_i(x)} \nabla c_i(x) \tag{2.16}$$

The best Lagrangian multipliers $z_i^*, i \in I$ can be obtained when x is near the minimizer $x(\mu)$ and the value of μ is not large. If the value of μ is not large, and x is near the point a for which f(x) > f(a) at all neighboring points x, then the ideal Lagrange multipliers $z_i^*, i \in I$ is approximately:

$$z_i^* \approx \mu / c_i(x), i \in I \qquad (2.17)$$

The conventional architecture for algorithms built around the primal log-barrier function is as below.

Algorithm 2.2 (Unconstrained Primal Barrier Method).

Let $\mu_0 > 0$, a series $\{\tau_i\}$ alongside $\tau_i \to 0$, and a initial point x_0^s;

do

Obtain the estimated minimizer x_i of $P(.; \mu_i)$, beginning at x_i^s,

and ending as soon as $\|\nabla P(x_i ; \mu_i)\| \leq \tau_i$;

Calculate the Lagrange multipliers z_i by (2.16);

if last test of convergence is fulfilled

stop at solution close to x_i ;

Select another criterion penalty $\mu_{i+1} < \mu_i$;

Select another initial point x_{i+1}^s;

while i > 0

end (do)

This class of Interior Point Method called primal barrier method was initially put forward by Frisck (1995) in the 1950s. The original barrier method was investigated by Fiacco and McCormick (1990), and they have made their own contribution towards the improvement and propagation of the barrier method. This approach however has become obsolete with the introduction and adoption of the Sequential Quadratic Programming (SQP)

due to several advantages it have over the barrier approach. The primal-dual Interior-Point method is also preferred to the barrier approach the minimizer $x(\mu)$ is very hard to find as $\mu \downarrow 0$ since the output of the function $P(x; \mu)$ is not directly to its input. The primal-dual Interior Point Method by solving a series of unconstrained optimisation problems which form a part of another more inclusive problem is able to proffer answer to the constricted optimisation problem

$$\min_{x \in R^N} (F_\mu(x) = F(x) + \pi B(x))$$

when the values of the barrier parameter μ decrease to 0. A secondary function B(x) which is also known as the barrier function is used in the unrestricted problem that is part of another problem. The logarithmic barrier is the most popular among the secondary functions.

$$B(x) = -\sum_{i=1}^{m} \log\big(c_i(x)\big),$$

let $c_c(x) = [c_x + p]_i$, the supplemented criterion F_μ turns out to be boundless at the border of the feasible area so that x^*_μ the minimizers satisfy $C_i(x^*_\mu) > 0$ for all i.

2.8.2 Primal-dual methods

It was rightly observed by Wright (1997) that the primal-dual methods are a modern class of interior-point methods that have been used to proffer solution to large-scale nonlinear optimisation problems. Primal-dual methods, unlike to what is obtainable in the conventional primal method, equally compute the primal variables x and dual Lagrange multipliers λ

correlating to the constraints simultaneously. The perturbed Karush-Kuhn-Tucker (KKT) equations below can be resolved through the exact primal-dual solution (x^*_μ, λ^*_μ) for a specified parameter μ

$$\begin{cases} \nabla F(x) - C^T\lambda = 0 \\ \lambda_i C_i(x) = \mu, i = 1, \dots, m \end{cases}$$

with the constraint $(C(x),\lambda) \geq 0$.

The Newton's algorithm and line search method are used to recursively solve any primal or primal-dual sub-problems for a given μ value (Boyd & Vandenberghe, 2004). Feasibility and convergence is imposed on the algorithm by judiciously choosing the size of step in the iteration. One of the predominant way to achieve this is to properly decrease the merit function used in evaluating the rate of progress towards the solution. The dual variables of the primal dual can be safeguarded through the use of $F\mu$ as a function that is capable of integrating the primal and dual variables (Armand, Gilbert & Jan-Jegou, 2000); and at the same time assesses the coherence between data and the suitable model for a specific selection of the parameters (Johnson, Seidel & Sofer, 2000). The key drawback of the barrier functions is their tendency to bring about ineffectiveness of traditional line exploration methods thereby necessitating the development of more efficient line search methods (Emilie, Saïd & Jérôme, 2011).

According to Carl (2006), the state-of-the-art Interior-Point algorithm have gained popularity as the choicest approach for providing solution to large scale linear programming problems. They are however limited due to their inability to solve problems that are unsteady in nature. This is because contemporary Interior-Point algorithm might not be able to cope

with the increasing need of the large number of constraints. Efforts to increase the efficiency of the Interior-Point algorithm have led to the development of another variant of this algorithm that can handle unsteady linear programming problems. These algorithms lower the number of work per iteration by using only small number of constraints thereby drastically decreasing their total processing time (Luke, 2010). The primal-dual interior-point (PDIP) algorithm is an excellent example of an algorithm that uses the constraint-reduction methods. Mehrotra (1992) in his research work developed the Mehrotra's Predictor-Corrector PDIP algorithm which has been executed in almost all the interior-point software suite for solving both linear and convex-conic problems (Frisch, 1995).

Strengths of primal-dual algorithm:

- It can efficiently handle large linear programming problems, and the bigger the problem size the more noticeable the efficiency of the primal-dual algorithm.
- The algorithm is not susceptible to degradation and the number of iterations does not depend on the number of vertices in the feasible search space.
- Primal-dual algorithm uses considerably less iteration compared to what we have in simplex method.
- The algorithm is able to get the idea solution for a linear programming problem in not more than 100 iterations irrespective of the huge number of variables involved in nearly all its implementations.

Shortcomings of primal-dual algorithm:

- Inability to detect the possibility of having unbounded status of the problem, up to some extent the primal-dual algorithm can be tagged as incomplete. Some researchers

have however been able to handle this problem through the use of model that are undiversified in nature (Wright, 1996) and (Quintana & Torres, 1997).

- The computational cost of each iteration of primal-dual algorithm is higher than that of the simplex algorithm. When we have a large linear programming problem that involves more than 100 variables, the primal-dual algorithms performs better than the simplex algorithm. This is due to the fact that the total work done in providing solution to a linear programming problem is the multiplication of the number of iterations and the work done during iteration.
- The primal step has the inclination of producing inferior steps that defile the boundaries $s > 0$ and $z > 0$ extensively, causing the progress to dwindle.

2.9 Feasible and Infeasible Interior-Point Methods

Assuming we have the problem formulated below

$$\min_{x,s} f(x) \qquad (2.1a)$$

$$\text{subject to} \quad C_E(\mathrm{x}) = 0 \qquad (2.1b)$$

$$C_I(\mathrm{x}) = 0 \qquad (2.1c)$$

let f, C_E and C_1 are adequately smooth functions of the variable $x \in R_n$. Where f is a scalar-valued function, and C_E and C_1 are vector-valued functions. A feasible method for (2.1) is the one in which the initial point and all successive iterates fulfill the inequality constraints (2.1c).

The realization of the inequality constraints during iteration is not made compulsory by the infeasible interior methods. The slack variable is used by the infeasible interior point method to convert (3:1) to the feasible equivalent of the problem.

$$\min_{x,s} f(x) \qquad (2.2a)$$

$$\text{subject to} \quad C_E(x) = 0 \qquad (2.2b)$$

$$C_I(x) - s = 0 \qquad (2.2c)$$

$$s \geq 0. \qquad (2.2d)$$

The interior point methods make use of the Newton's method during iteration to provide solution that is accurate to a greater extent to the barrier problem.

Algorithm 2.3: Generic Infeasible Interior Point Algorithm

Given that x (probably infeasible) is a iterate and s > 0 is a slack vector

Do

Calculate the step dist = ($dist_x$; $dist_s$):

Define the testing point $x_{testing} = x + dist_x$; $s_{testing} = s + dist_s$:

Do

Calculate a smaller step dist:

Let $x_{testing} = x + dist_x$; $s_{testing} = s + dist_s$:

while $\Phi(x_{testing}; s_{testing})$ is not satisfactorily lesser than $\Phi(x; s)$

Let $x_+ = x_{testing}$; $s_+ = s_{testing}$:

while the rule for ending is not yet fulfilled

When the technique in use is the trust region method, reducing the radius of the trust region and re-calculating a step will enable us to get a smaller step. For line search method, using a retrace your steps line search would produce the same result.

If the present iterate is precisely feasible with respect to the inequality constraints, some alterations can be made to make sure that the next iterate is also feasible. The slack variable can be redefined once the step $dist_x$ and $dist_s$ have been computed as $s_{testing} \leftarrow g(x_{testing})$. Analysis is then carried out to see if the point $(x_{testing}; s_{testing})$ is suitable for the merit function. The step is discarded and a fresh, experimental step which is shorter is calculated if the present step is not satisfactory. The infeasible Generic Algorithm is first employed when the first iterate x_0 does not suit all the constraints in the inequality. This continues until all inequalities are more than certain positive maximum value. When such happens, the algorithm swaps to the feasible form and for the remaining part of the optimisation process continues to be feasible. The algorithm is below:

Algorithm 2.4: Feasible-Reset Interior Point Algorithm

Given that x (probably infeasible) is a iterate and $s > 0$ is a slack vector, and γ a positive maximum value

if $g(x_{testing}) < \gamma e$ **then**

Execute infeasible Generic Interior Point Algorithm until $g(x_{testing}) \geq \gamma e$

Set $s = g(x_{testing})$

end if

do

Calculate the step $dist = (dist_x; dist_s)$:

Specify the experimental point $x_{testing} = x + dist_x$; $s_{testing} = s + dist_s$:

do

Calculate a smaller step dist:

Let $x_{testing} = x + dist_x$; $s_{testing} = s + dist_s$:

while $\Phi(x_{testing}; s_{testing})$ is not satisfactorily lesser than $\Phi(x; s)$

Let $x_+ = x_{testing}$; $s_+ = s_{testing}$:

while a stopping test is not yet fulfilled

Other condition that does not treat each constraint equally and that put into consideration the range of the constraints can be applied to take the place of the test g (xtesting) ≥ ɤe. It is worth noticing that the vector dists is not necessary in Feasible-Reset Algorithm. Richard, Jorge, and Richard (2005) presumed that during iteration there is calculation of a phase in the slacks and variables thereby making both the feasible and infeasible genres to need similar data structure and variables. It is advantageous in several applications that all iterates produced by an optimisation algorithm be feasible as regards some or all of the inequality constraints. An inherent structure for developing feasible algorithms is supported by the interior point method (Jorge & Stephen, 2006).

2.10 Line Search Interior-Point Method

Interior point algorithms employ the reduction of rule boosted by a barrier function to guarantee the satisfaction of the constraints. The existence of barrier function will however make the convergence speed of the iterative succession algorithm to be very slow when multifunctional line search methods are used. An elaborate description of a line search interior-point method is given in the algorithm below. The orientation end product of the merit function ϕ_v at (x, s) is given by $D\phi(x, s; p)$ in the direction P. The stopping conditions are based on the error function.

Algorithm 2.5: (Line Search Interior-Point Algorithm).

Select x_0 and $s_0 > 0$, and compute initial values for the multipliers y_0 and $z_0 > 0$.

If a quasi-Newton approach is used, choose an $n \times n$ symmetric and positive definite initial matrix B_0. Choose an original barrier parameter $\mu > 0$, parameters $\eta, \sigma \in (0, 1)$, and tolerances $\in_\mu$ and $\in_{TOL}$. Set $k \leftarrow 0$.

repeat until $E(x_k , s_k , y_k , z_k; 0) \leq \in_{TOL}$

repeat until $E(x_k , s_k , y_k, z_k; \mu) \leq \in_\mu$

Calculate the primal-dual direction $p = (p_x, p_s, p_y, p_z)$

Calculate α_s^{max}, α_z^{max}

Let $p_w = (p_x , p_s)$;

Calculate step lengths $\propto_s$, $\propto_z$ meeting the necessary conditions

Calculate $(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1})$

if a quasi-Newton approach is used

update the *estimate* B_k;

Set $k \leftarrow k + 1$;

end

Set $\mu \leftarrow \sigma\mu$ and update $\in_\mu$;

end

The strength of the barrier can be described as $\in_\mu = \mu$, as in the above Algorithm. A flexible approach that is up to date information at every step about μ which is the barrier parameter can easily be executed using this approach. A second-order rectification or an approach that is not monotonic ought to be executed to avoid the merit function triggering the Maratos effect. We can abandon the merit function for a filter method to execute the line search. This will guarantee the global convergence of the algorithm (Jorge & Stephen, 2006).

2.11 Trust-Region Interior-Point Method

This is a kind of interior-point method that uses the trust regions to ensure convergence. The design of the trust-region tolerates unlimited autonomy in the selection of the Hessian. It also supports a system for handling individualities of the Jacobian and Hessian. This is ensured by a tradeoff between flexibility and complexity in iteration compared to the line search approach.

The interior-point described below can be directly compared to the line search method. The two of them are however not the same as the trust region method is not completely a primal-dual method, as the first thing it does is to calculate the variables' steps (x, s) before it computes the current estimated value of the multipliers. Moreover, the approach of the trust-region restricts movements in the direction of the border of the feasible region by ordering the variables. The advantage of this approach is that it will make the algorithm to produce steps that divers from that of the line search approach, and having good convergence properties. There are two types of trust region algorithm that will be described in this section. They include: Trust-Region Interior-Point Method for Barrier Problem, and Trust-Region Interior-Point Method for Nonlinear Programming (Jorge & Stephen, 2006).

2.11.1 Trust-Region Interior-Point Method for Barrier Problem

The combination of trust regions and sequential quadratic programming (SQP) technique can be used to provide solution to the barrier problem (2.6). The SQP method when applied to solve the barrier problem usually produces incompetent steps which are against the affirmativeness of the variables that are added to a constraint to turn the inequality into an equation. The steps produced by the SQP are also interrupted before its planned end by the trust region constraint. Jorge & Stephen (2006) developed an SQP technique customized to the barrier problems to surmount these disadvantages of SQP. The first thing they did was

to calculate the approximate Lagrange multiplier (y, z) and then calculate step $p = (p_x, p_s)$ given the iterate (x, s) for a given barrier parameter μ. This provides an estimated solution to the problem below:

$$\min_{Px.Ps} \nabla f^T p_x + \tfrac{1}{2} P_x^T \nabla_{xx}^2 - \mu e^T s^{-1} p_s + \tfrac{1}{2} p_s^T \sum p_s \qquad (2.18a)$$

$$\text{Subject to } A_E(x)P_x + C_E(x) = r_E, \qquad (2.18b)$$

$$A_I(x)P_x - P_s + (C_I(x) - s) = r_I, \qquad (2.18c)$$

$$\|(P_x, s^{-1}P_s)\|_2 \leq \Delta, \qquad (2.18d)$$

$$P_s \geq -rs. \qquad (2.18e)$$

Where

$\sum$ = Primal-dual matrix (2.12),

τ = Scalar and $\tau \in (0, 1)$ is made to have a value closer to 1 (e.g. 0.997),

S^{-1} = Scaling

P_s = Step vector

r_E and r_I. = Relaxation vectors

According to Jorge & Stephen (2006), the function carried out by (2.18e) is similar to the one of the threshold principle in (2.9). Dogmatically following the conventional approach of setting $r = (r_E, r_I) = 0$, however this can make the constraints (2.18b) – (2.18d) to be unsuitable or to present a step p that cannot effectively advance to viability. They proved further that the calculated step of (2.18) have some correlation with the line search step of primal-dual.

Jorge & Stephen (2006) observed that the constraint (2.18d) of the trust-region gives the assurance that the problem (2.18) can be solved predictably. Though $\nabla_{xx}^2 L$ (x, s, y, z)

might not be convincingly positive, it will still produce the desired solution and there will be no need to amend the Hessian. Another notable advantage of the trust region is that it ensures that it guarantees that sufficient advancement is achieved at every iteration. To make the effect of the scaling S^{-1} vivid, the structure of the trust region need to plan for the possibility of the prerequisite that the slack variables do not get to zero before the due time. This is the rationale behind the introduction of the scaling S^{-1} so that it can constrain the elements i of the step vector P_s where s_i is not far from zero which is its lower boundary. The algorithm below is the Trust-Region Algorithm for Barrier Problems:

Algorithm 2.6: (Trust-Region Algorithm for Barrier Problems).

Input parameters: $\mu > 0$, $x0$, $s0 > 0$, μ, and $\_0 > 0$.

Calculate Lagrange multiplier estimates y_0 and $z_0 > 0$.

Let $k \leftarrow 0$.

do

Calculate $p = (p_x, p_s)$ by approximately solving (2.18).

if p offers adequate reduction in the merit function φ_v

Let $x_{k+1} \leftarrow x_k + p_x$, $s_{k+1} \leftarrow s_k + p_s$;

Calculate additional multiplier estimates y_{k+1}, $z_{k+1} > 0$

and let $\nabla_{k+1} \geq \nabla_k$;

else

Define $x_{k+1} \leftarrow x_k$, $s_{k+1} \leftarrow s_k$, and set $\nabla_{k+1} < \nabla_k$;

end

Set $k \leftarrow k + 1$;

until $E(x_k, s_k, y_k, z_k ; \mu) \leq \in_\mu$

end

The above algorithm is relevant for a predetermined value of the barrier parameter μ. A comprehensive interior-point algorithm propelled by a sequence $\{\mu k\} \rightarrow 0$ is explained in the next section.

2.11.2 Trust-Region Interior-Point Method for Nonlinear Programming

The trust-region interior-point algorithm can also be used for providing solution to the nonlinear programming problem (2.1). The Fiacco–McCormick approach for bringing up to date the barrier parameter was used by stating the ending requirements in terms of the error function E. In a quasi-Newton approach, The Hessian $\nabla_{xx}^2 L$ is substituted with a proportional estimation.

Algorithm 2.7: Trust-Region Interior-Point Algorithm

Select a value for the parameters $\eta > 0$, $\tau \in (0, 1)$, $\sigma \in (0, 1)$, and $\zeta \in (0, 1)$,

Pick the halting tolerances μ and TOL.

If a quasi-Newton approach is used, select an $n \times n$ symmetric initial matrix $B0$.

Choose initial values for $\mu > 0$, $x0$, $s0 > 0$, and 0.

Let $k \leftarrow 0$.

do

do

Calculate Lagrange multipliers;

Calculate $\nabla^2_{xx} L(x_k, s_k, y_k, z_k)$ or update a quasi-Newton approximation B_k, and define $\sum_k$ by (2.12);

Calculate the normal step $v_k = (v_x, v_s)$;

Calculate $\tilde{p}_k$ by using the estimated CG approach;

Find the sum step p_k;

Update v_k to fulfill the needed condition;

Calculate $\text{pred}_k(pk)$;

if $\text{ared}_k(pk) \geq \eta\, \text{pred}_k(pk)$

Let $x_{k+1} \leftarrow x_k + p_x$, $s_{k+1} \leftarrow s_k + p_s$;

Select $\Delta_{k+1} \geq \Delta_k$;

else

set $x_{k+1} = x_k$, $s_{k+1} = s_k$; and select $\Delta k+1 < \Delta k$;

endif

Let $k \leftarrow k + 1$;

until $E(xk, sk, yk, zk; \mu) \leq \mu$

Set $\mu \leftarrow \sigma\mu$ and update ϵ_μ ;

until $E(xk, sk, yk, zk; 0) \leq$ TOL

end

This algorithm still suffers from the Maratos effect which can make it to disallow steps that cause beneficial advance to a solution. According to Jorge & Stephen (2006), this shortcoming have however been conquered through careful usage of a second-order amendment step.

2.12 Summary of Literature Review

In this chapter, we did discussed the characteristics of a swarm robotic system such as large number of autonomous robots; ability to detect and communicate from one robot to the other within the same local neighbourhood; distributed system that is not dependent on global information; and cooperative behavior, and robot to robot communication and robots to environment communication. The advantages and disadvantages of swarm robotics were examined.

A comprehensive review of the different swarm robotic algorithms was been done in this chapter. We also did a comparative review of foregoing research in the field of particle swarm optimisation. Variants of PSO and how they have been applied to swarm robotics was also presented. The comprehensive discussion of the applicability of PSO as an optimisation algorithm to support swarm robotics (or multi-objective dynamic optimisation in general) as presented in this chapter exposed critical limitations that describes the recent trends of customizing a new swarm algorithms for each swarm robotic project. These limitations can be addressed with the introduction of some other technique to improve the performance of PSO so that it can tackle dynamic optimisation problems adequately. There is the need to address some of the problem associated with the existing variants of PSO that we have discussed such as premature convergence, and the challenge of some of the particles been trapped in the local minima. An exhaustive review of the literatures accessible in this subject is presented to give a broad understanding of this topic.

Furthermore, we did a holistic overview of the various Interior Point method algorithms. We considered the primal-dual interior-point (PDIP) method is as an excellent example of an algorithm that uses the constraint-reduction methods. The primal-dual methods were classified as a new category of interior-point methods that have of recent been practically employed for solving large-scale nonlinear optimisation problems. We discussed the major setback of the barrier functions as the ineffectiveness of traditional line exploration methods thereby necessitating the development of more efficient line search (Chouzenoux, Idier & Moussaoui, 2011). We also mentioned that primal-dual method can efficiently handle large linear programming problems. The bigger the problem size the more noticeable the efficiency of the primal-dual algorithm. The algorithm is not susceptible to degradation and the number of iterations does not depend on the number of vertices in the feasible search space (Wehenkel & Glavic, 2004). The drawbacks of the primal-dual was explained such as its inability to detect the possibility of having unbounded status of the problem, and the high computational cost per each iteration. In the next chapter we will do a survey on PSO algorithms using different benchmarking functions.

CHAPTER 3: A SURVEY OF PERFORMANCE OF PARTICLE SWARM OPTIMISATION (PSO) ALGORITHM ON BENCHMARK PROBLEMS

3.1 Introduction

It is evident from the previous chapter, that in order to apply PSO into any swarm robotics implementation, a high performing PSO algorithm is need thereby necessitating the development of more efficient PSO variants and thus contributing towards the ever-expanding pool of PSO algorithm. This should have not been the norm since the natural characteristics of the algorithm should be able to support any swarm robotics project. Apparently, the algorithm is ill suited with some fundamental problems. This study is aimed towards identifying the performance of the conventional PSO algorithms, and from this comparative review, work on addressing issues related to swarm robotic applications towards the creation of a generic PSO, adaptable to any swarm robotic project.

We carried out certain experiments using different benchmark functions. The purpose of this experiment is firstly to establish the ground truth of these existing algorithms, and to determine if they are functioning as described in the literature. Secondly, to determine the global optimum and local optimal of each of the three PSO variants under the different benchmarking functions, and to confirm the problem of the particles in PSO been trapped in local optima. Lastly, we are validating the existence of the premature convergence problem of PSO. In general, we wanted to investigate the performances of the three PSO variants using the global optimum and local optimal using the standard benchmark functions. Moreover we wanted to examine their convergence properties through these benchmark functions.

Our intention is to develop a single generic PSO that would be applicable for swarm robotics tasks. Though there are many variants of PSO that have been applied to swarm robotics, these variants can categorically be grouped into three main versions based on their properties (i.e., minor fundamental changes of each from the basic PSO) of the PSO algorithm. The three variants are the original PSO, synchronous PSO (SPSO), and asynchronous PSO (APSO).

3.2 Benchmark functions

Sixteen benchmark functions were selected. They can be classified as Unimodal or Multimodal, and either Static or Dynamic functions. The selected functions are *Sphere* (Bijaka, Yuhui and Meng-Hiot 2011), *Alpine* (Clerc 2004), *DeJong f3* (Xin et. al. 2004), *DeJong f4* (Xin et. al. 2004), *Foxhole* (Grana et. al. 2004), *Tripod* (Clerc 2004), *NDParabola* (Clerc 2004), *Griewank* (Dervis and Bahriye 2007), *Rastrigin* (Dervis and Bahriye 2007), *Rosenbrock* (Dervis and Bahriye 2007), *Ackley* (Dervis and Bahriye 2007), *Shaffer f6* (Sun, Lai and Wu 2012), *Shaffer f6 modified* (Matlab Central 2013), *f6 Linear Dynamic* (Matlab Central 2013), *f6 Bubble Dynamic* (Matlab Central 2013), and *Shaffer f6 Spiral Dynamic* (Matlab Central 2013). The chosen functions have been selected to test the ability of PSO and its variants to escape premature convergence. Moreover, some of the functions test the capability of the PSO algorithms to escape local minima, while the multimodal functions that were used test the performance of PSO algorithms in a dynamic environment. All functions used for the experiment are for minimization problems and their properties are outlined in the tables below.

Table 3.1: Benchmark Functions and their Mathematical Equation.

Function Name	**Mathematical Equation**
Sphere (De Jong f1)	$f(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock (De Jong f2)	$f(x) = \sum_{i=1}^{N} 100\ (x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{N} cos\ \frac{xi}{\sqrt{i}} + 1$
Rastrigin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$
Ackley	$f(x) = -20 * e^{-0.2} \sqrt{\frac{1}{N} \sum_{i=1}^{N} x_i^2} - e^{\frac{1}{N}} \sum_{i=1}^{N} \cos\ (2\pi x_i + 20 + e$
Alpine	$f(x_d) = \sum \lvert x_d \sin(x_d) + 0.1\ x_d \rvert$ where d = 1, 2, 3
De Jong f3	$f(x) = \sum_{i=1}^{5} integer(x_i)$
De Jong f4	$f(x) = \sum_{i=1}^{30} i * x_i^4$
Shekel's foxhole	$f(x) = -\sum_{i=1}^{m} (\sum_{j=1}^{n} [(x_j\ a_{ij})^2 + c_j])^{-1}$ where c_j, i=1,…,m), (a_{ij}, j = 1,…,n, i= 1,…,m) are constant numbers fixed in advance.

NDParabola	$f(x) = \sum_{d=1}^{D} x_d^2$
Schaffer f6	$f(x) = 0.5 + \frac{(sin^2\sqrt{x^2 + y^2})}{(1 + 0.001\,(x^2 + y^2))^2} - 0.5$
Schaffer f6 modified	$f(x) = 0.5 + \frac{(sin^2\sqrt{x^2 + y^2})}{(1 + 0.001\,(x^2 + y^2))^2} - 0.5$
Tripod	$f(x) = p(x_2)\big(1 + p(x_1)\big) + \lvert x_1 + 50p(x_2)\big(1 - 2p(x_1)\big)\rvert + \lvert x_2 + 50\big(1 - 2p(x_2)\big)\rvert$
F6 Linear Dynamic	$f(x) = 0.5 + \frac{\sin(\sqrt{(x - x_{centre})^2 + (y - y_{centre})^2)^2}}{(1 + 0.01((x - x_{centre})^2 + (y - y_{centre})^2)^2)} - 0.5$
F6 Bubbles Dynamic	$f(x) = 2 * \left(\left(0.5 + \frac{\sin(\sqrt{(x - x_1)^2 + (y - y_1)^2)^2}}{(1 + 0.01((x - x_1)^2 + (y - y_1)^2)^2)} - 0.5\right) * \left\lvert sin\left(\frac{cputime}{10} - cputime\right) + \frac{\pi}{2}\right\rvert\right) + \left(0.5 + \frac{\sin(\sqrt{(x - x_2)^2 + (y - y_2)^2)^2}}{(1 + 0.01((x - x_2)^2 + (y - y_2)^2)^2)} - 0.5\right) * \left\lvert \sin\left(\frac{cputime}{10} - cputime\right) + \frac{\pi}{2}\right\rvert) - \left(\left(0.5 + \frac{\sin(\sqrt{(10000 - x_1)^2 + (10000 - y_1)^2)^2}}{(1 + 0.01((10000 - x_1)^2 + (10000 - y_1)^2)^2)} - 0.5\right) * \left\lvert sin\left(\frac{cputime}{10} - cputime\right) + \frac{\pi}{2}\right\rvert\right) + \left(\left(0.5 + \frac{\sin(\sqrt{(10000 - x_2)^2 + (10000 - y_2)^2)^2}}{(1 + 0.01((10000 - x_2)^2 + (10000 - y_2)^2)^2)} - 0.5\right) * \left\lvert sin\left(\frac{cputime}{10} - cputime\right) + \frac{\pi}{2}\right\rvert\right)$
F6 Spiral Dynamic	$f(x) = 0.5 + \frac{\sin(\sqrt{(x - x_{centre})^2 + (y - y_{centre})^2)^2}}{(1 + 0.01((x - x_{centre})^2 + (y - y_{centre})^2)^2)} - 0.5$

Table 3.2: Parameters for Test Functions

Function Name	**Properties**	**Dimension**	**Initial Range**	**Global Optima (x*)**
Sphere (De Jong f1)	Is a unimodal function, it is simple, there is no communication between its variables.	30	$[-100; 100]^n$	[0,0,…,0]
Rosenbrock (De Jong f2)	It is a unimodal function, have complicated landscape due to very narrow ridge.	30	$[30; 30]^n$	[1,1,…,1]
Griewank	Non-linear multimodal function. Highly multimodal due to the addition of the cosine modulation that produces many widespread local minima.	30	$[-600;600]^n$	[0,0,…,0]
Rastrigin	This is a multi-modal version of the sphere function with the addition of cosine modulation to produce frequent local minima. It contains millions of local optima.	30	$[-5.12; 5.12]^n$	[0,0,…,0]
Ackley	Multi-modal function with deep local minima. It has several local minima.	30	$[-32;32]^n$	[0,0,…,0]
Alpine	It has many local and global minima of value zero.	3		[0,0,…,0]
De Jong f3	A uniformly increasing stepping function in five dimensions.	5	$[-5.12; 5.12]^n$	[0,0,…,0]
De Jong f4	Is a noisy function.		$[-1.28; 1.28]^n$	[0,0,…,0]
Shekel's foxhole	Is a multimodal test function.		[-65.538; 65.538]	[1,1,…,1]
NDParabola	Was used to test for global minimization problems in Clerc's "semi-continuous challenge." It works very well with gradient	30	[-20;20]	[0,0,…,0]

	methods, but presents a challenge for PSO which is a stochastic method.			
Schaffer f6	Is a complex multimodal function. Most hill-climbing and reactive search methods find it very difficult due to its circular local maxima. It is considered a GA-hard function to optimize.		[-100,100]	[0,0,…,0]
Schaffer f6 modified	This is the sum of five (5) Schaffer f6 functions with different centres to look for local minimum.		[-100,100]	[0,0,…,0]
Tripod	A semi-continuous function. This function presents a problem that many algorithms such as GA and PSO that are easily trapped in one of the two local optima find very difficult to cope with.	2	[-100,100]	[0,-50]
F6 Linear Dynamic	This is a version of Schaffer f6 that moves the optima minimum linearly along a 45 degree angle in x, y space.		[-100,100]	
F6 Bubbles Dynamic	This benchmark is made up of Schaffer f6 in which each goes on bubbles magnitude cycles up and down. They are 180 degree out of phase with each other.		[-100,100]	[-8,-8] and others at [8,8]
F6 Spiral Dynamic	This version of Schaffer f6 moves the minimum about a Fermat spiral according to the equation: r = a*(theta^2). Where theta is a function of time and is checked internally. x_{centre} = r (cos(theta)) y_{centre} = r (sin(theta))		[-100,100]	

Table 3.3: Parameters Settings of PSO variants

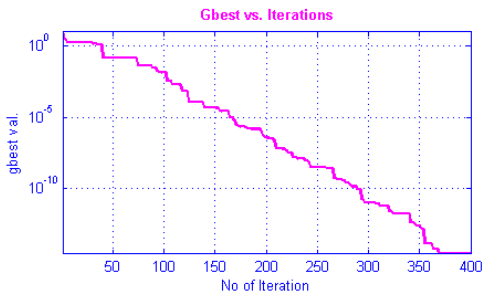
Algorithm	**Parameter Settings**
PSO	Population Size = 30, Dimension = 10, C1, c2 = 2, W = 0.9 to 0.4
SPSO	Population Size = 30, Dimension = 10, C1, c2 = 1.49, W = 0.9 to 0.4
APSO	Population Size = 30, Dimension = 10, C1, c2 = 1.49, W = 0.9 to 0.4

3.3 Results and Discussion

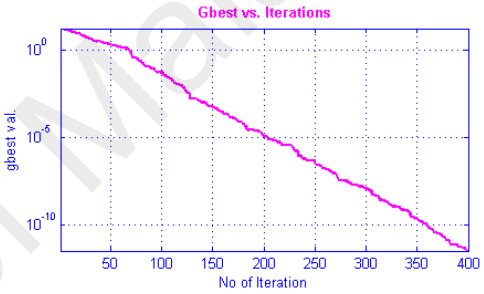
To affirm the existence of premature convergence, inability to escape being trapped in local minima, and the unsuitability of PSO algorithms and some of its variants for handling dynamic tasks we experimented with the original PSO, synchronous PSO (SPSO), and asynchronous PSO (APSO).algorithms. In general, we wanted to investigate the performances of the three PSO variants using the global optimum and local optimal using the standard benchmark functions. Moreover we wanted to examine their convergence properties through these benchmark functions. We have stated our intention of developing a single generic PSO that would be applicable for swarm robotics. Though there are many variants of PSO that have been applied to swarm robotics, these variants can be categorically group into three main versions based on their properties (i.e., minor fundamental changes of each from the basic PSO) of the PSO algorithm. The three variants are the original PSO, synchronous PSO (SPSO), and asynchronous PSO (APSO). We run each algorithm independently for 20

trials. Table 3.3 shows the settings of population size, dimension of problem, the cognitive and social scaling factors (c1 and c2), and the inertia weight *w* which is decreased linearly from 0.9 to 0.4 during the iterations. The results of the experiment for the 16-benchmark functions on the three variants of the PSO algorithm are depicted in Figure 3.3.1. Each figure represents the performance of the variants in solving each benchmark function. The graphs are generated during the simulation process and are saved as .jpg file from Matlab.

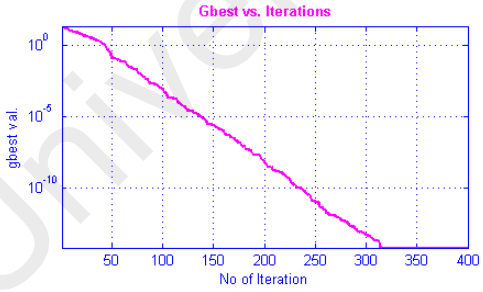
3.3.1 Simulation Results



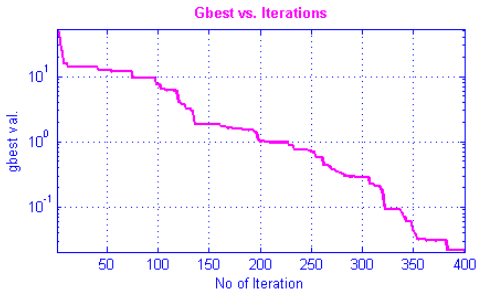
(a) Ackley function for PSO



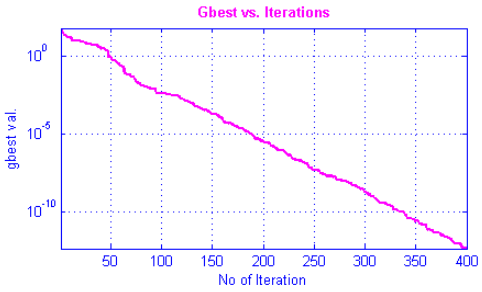
(b) Ackley function for SPSO



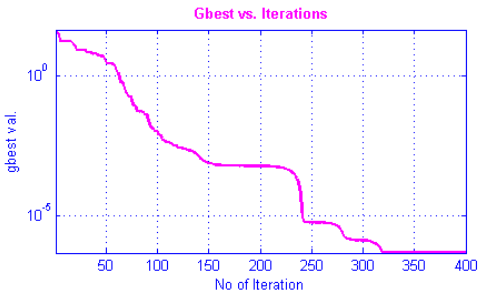
(c) Ackley function for APSO



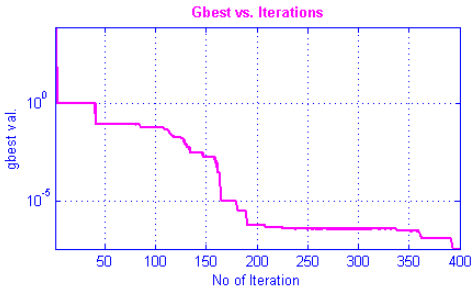
(d) Alpine function for PSO



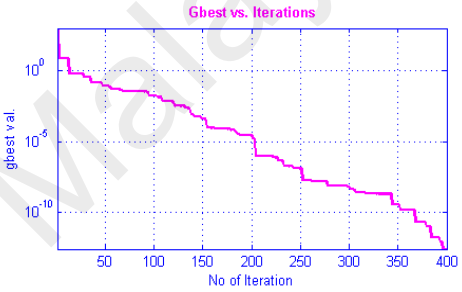
(e) Alpine function for SPSO



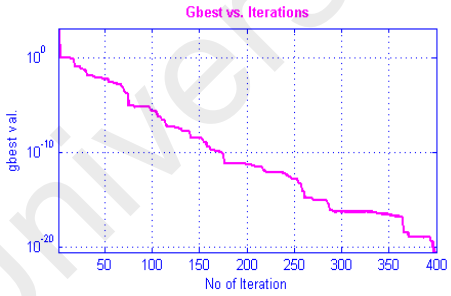
(f) Alpine function for APSO



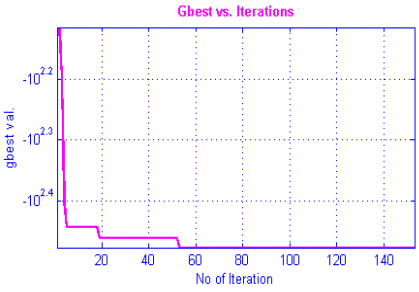
(g) Dejong f2 function for PSO



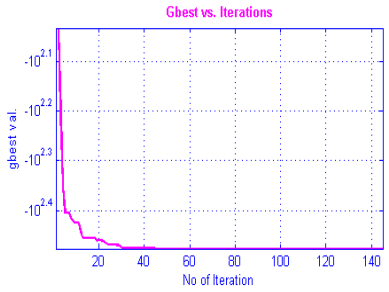
(h) Dejong f2 function for SPSO



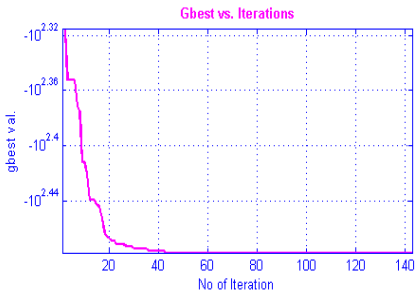
(i) Dejong f2 function for APSO



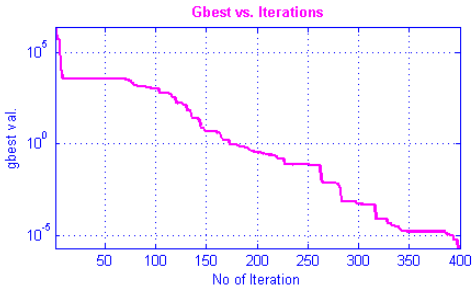
(j) Dejong f3 function for PSO



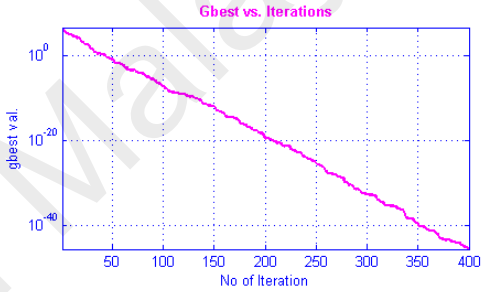
(k) Dejong f3 function for SPSO



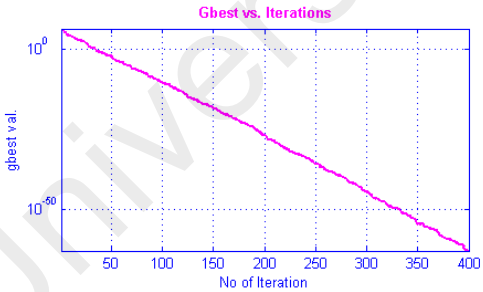
(l) Dejong f3 function for APSO



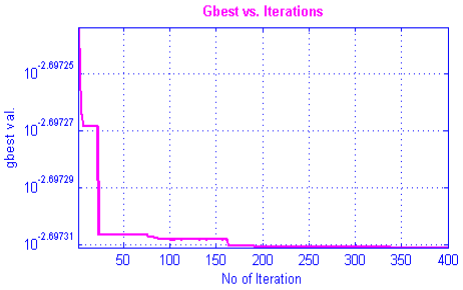
(m) Dejong f4 function for PSO



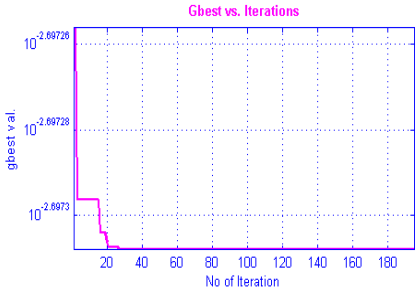
(n) Dejong f4 function for SPSO



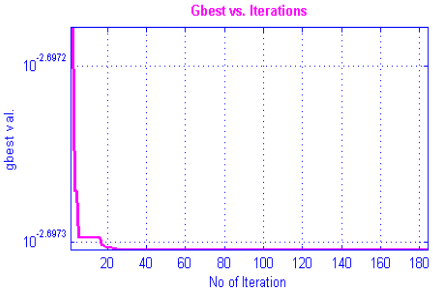
(o) Dejong f4 function for APSO



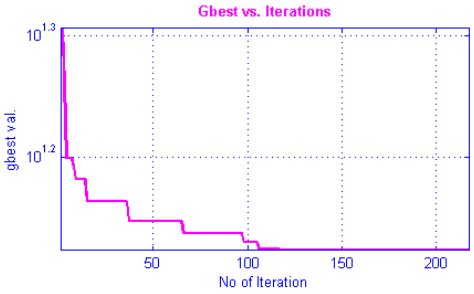
(p) Shekel foxhole function for PSO



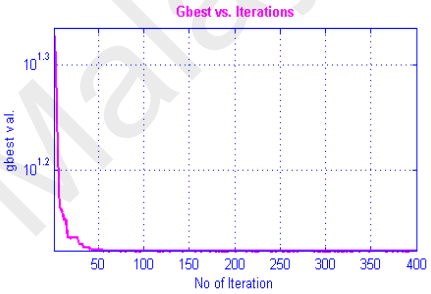
(q) Shekel foxhole function for SPSO



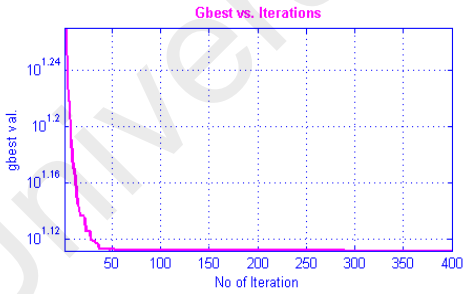
(r) Shekel foxhole function for APSO



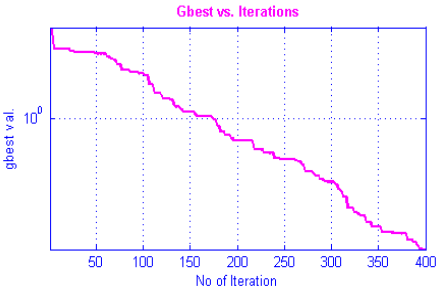
(s) Griewank function for PSO



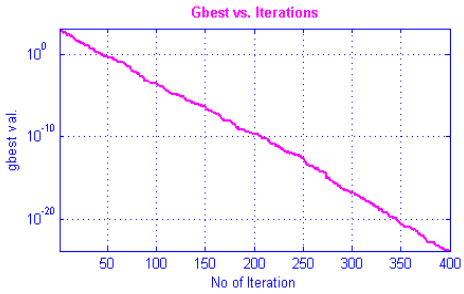
(t) Griewank function for SPSO



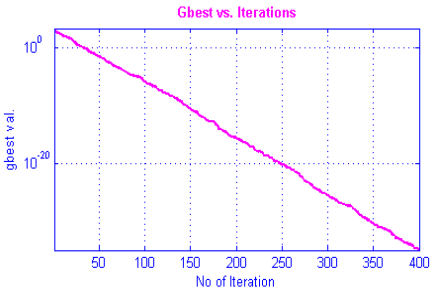
(u) Griewank function for APSO



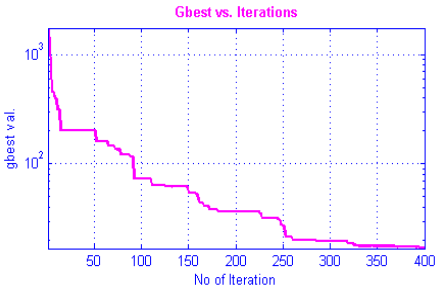
(v) NDParabola function for PSO



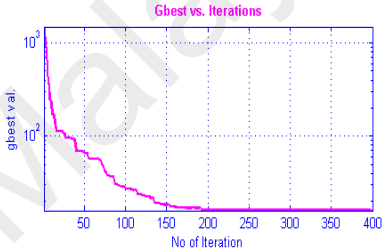
(w) NDParabola function for SPSO



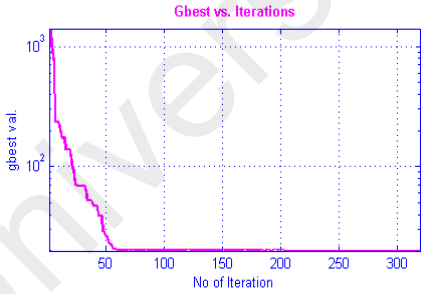
(x) NDParabola function for APSO



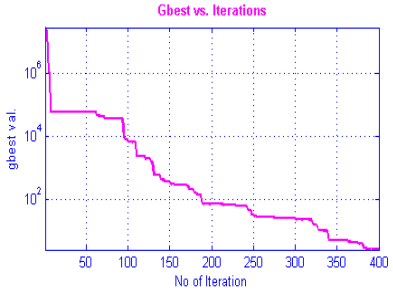
(y) Rastrigin function for PSO



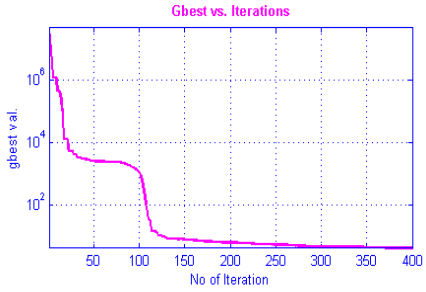
(z) Rastrigin function for SPSO



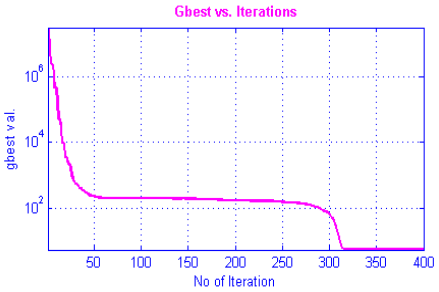
(aa) Rastrigin function for APSO



(ab) Rosenbrock function for PSO

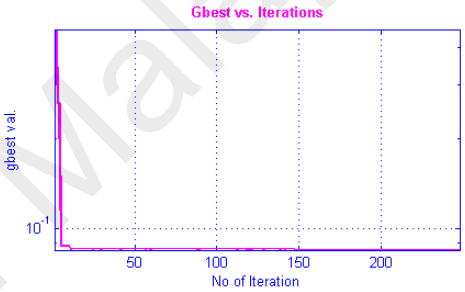


(ac) Rosenbrock function for SPSO

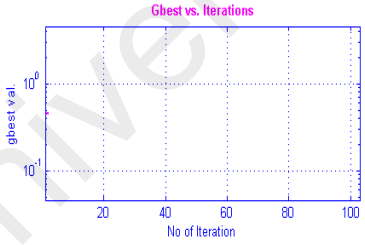


(ad) Rosenbrock function for APSO

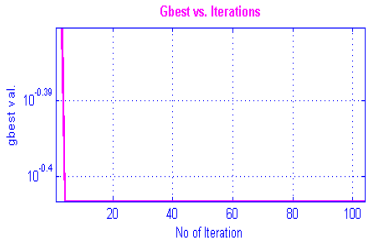
(ae) Schaffer f6 function for PSO



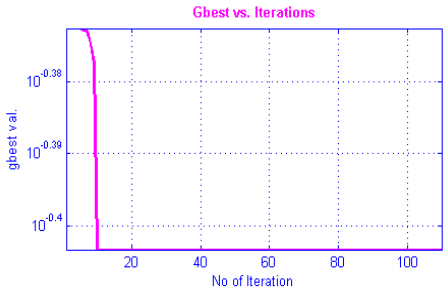
(af) Schaffer f6 function for SPSO



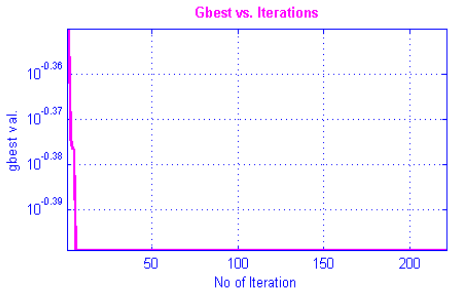
(ag) Schaffer f6 function for APSO



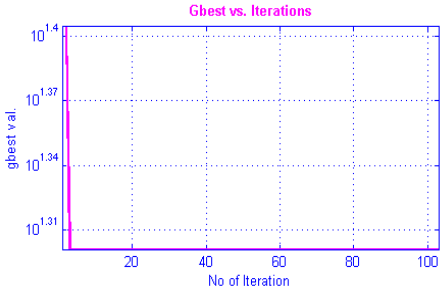
(ah) Schaffer f6 modified function for PSO



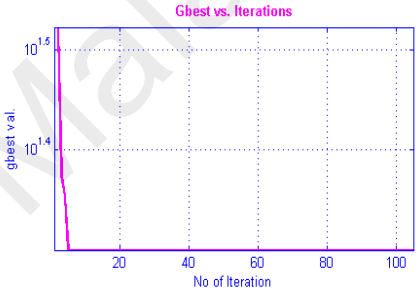
(ai) Schaffer f6 modified function for SPSO



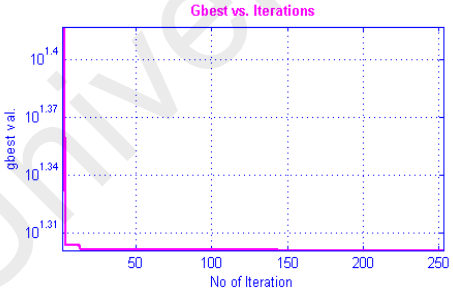
(aj) Schaffer f6 modified function for APSO



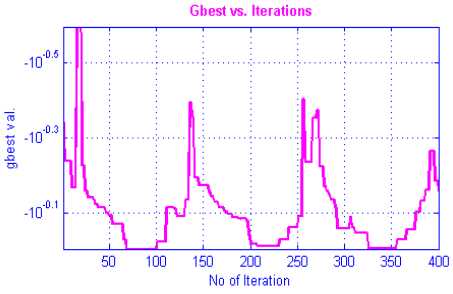
(ak) Tripod function for PSO



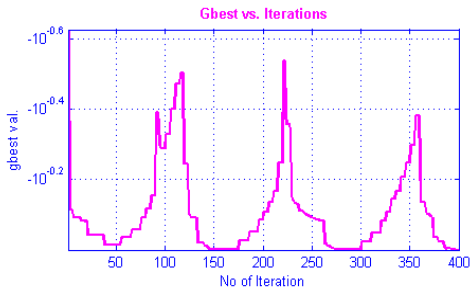
(al) Tripod function for SPSO



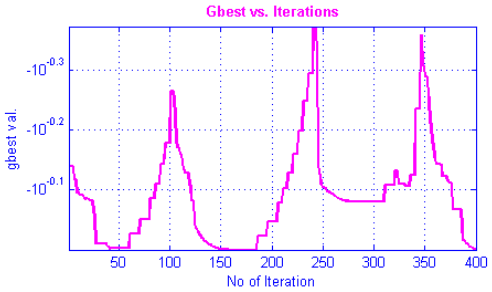
(am) Tripod function for APSO



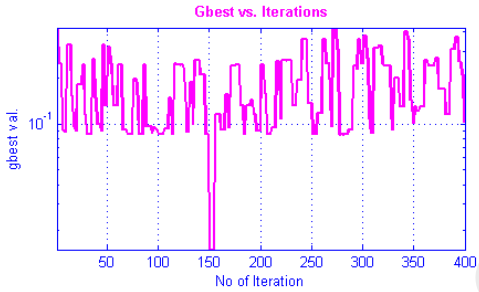
(an) Schaffer f6 Bubbles Dynamic function for PSO



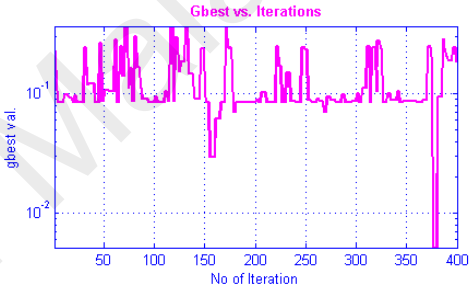
(ao) Schaffer f6 Bubbles Dynamic function for SPSO



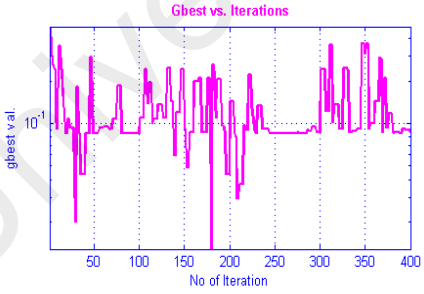
(ap) Schaffer f6 Bubbles Dynamic function for APSO



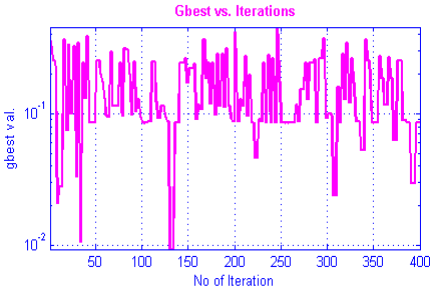
(aq) Schaffer f6 Linear Dynamic function for PSO



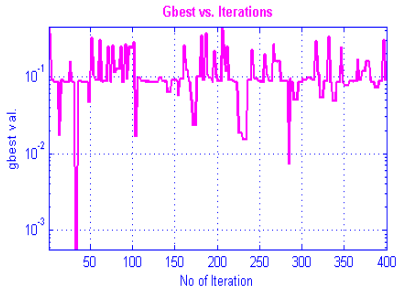
(ar) Schaffer f6 Linear Dynamic function for SPSO



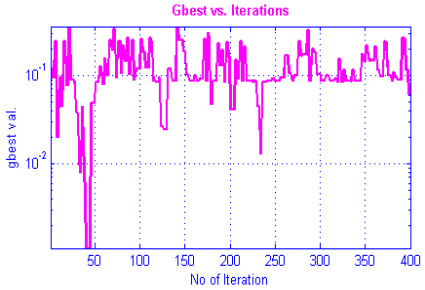
(as) Schaffer f6 Linear Dynamic function for APSO



(at) Schaffer f6 Spiral Dynamic function for PSO



(au) Schaffer f6 Spiral Dynamic function for SPSO



(av) Schaffer f6 Spiral Dynamic function for APSO

Figure 3.1: Simulation results of benchmark functions for PSO, SPSO, and APSO

3.3.2 Discussion

In the figure 3.1 above (a) – (c) is the Ackley function for PSO, SPSO and APSO respectively. There are many local minima produced by the function. We have global optimum for PSO at the 370th iteration, while SPSO have global optimum at the 400th iteration and APSO reached the global optimum at the 320th iteration. APSO therefore have a faster convergence speed than SPSO and PSO. The (d) – (f) above is the Alpine function for PSO, SPSO and APSO respectively. There are many local minima in the function. PSO reached the global optimum at the 380th iteration, while SPSO global optimum was reached at the 400th iteration and APSO has its global optimum at the 320th iteration. This further proves the superiority of the APSO over the SPSO and the PSO algorithms. The (g) – (i) above is the Dejong f2 function for PSO, SPSO and APSO respectively. We observed from the simulation result that there was a sharp drop in the value of the *gbest* as the iteration progresses, but convergence to the global optimum was difficult for PSO, SPSO and APSO. This might be the reason why this function was considered suitable for testing the

performance of optimisation algorithms. The (j) – (l) in figure 3.1 above is the Dejong f3 function for PSO, SPSO and APSO respectively. There was convergence to the global optimum at the 52nd iteration for PSO, while SPSO reached the global optimum at the 30th iteration. APSO has its global optimum at the 42nd iteration. SPSO performs slightly better than APSO and PSO with this function.

The (m) – (o) in the above figure is the Dejong f4 function for PSO, SPSO and APSO respectively. There are some local minima for PSO but convergence to global optimum was very difficult. SPSO and APSO do not have any local minima and there was no convergence to the global optimum. This is because the particles have been trapped in the local minima. The (p) – (r) in figure above is the Shekel foxhole function for PSO, SPSO and APSO respectively. There are many local minima for PSO, SPSO and APSO. The global optimum for PSO was reached at the 190th iteration, for SPSO it was at the 28th iteration while the global optimum was reached at the 22nd iteration for APSO. The (s) – (u) above is the Griewank function for PSO, SPSO and APSO respectively. PSO has many local optima and its convergence to global optima was at the 115th iteration. SPSO reached global optimum at the 48th iteration and APSO's global optimum was reached at the 42nd iteration.

The (v) – (x) above is the NDParabola function for PSO, SPSO and APSO respectively. There was no convergence to global optima for all the 3 algorithms used in this experiment. This is because the particles have been trapped in the local minima. This function poses a challenge to stochastic algorithms such as PSO and its variants but it has been discovered that it works perfectly with gradient methods. The (y) – (aa) in figure 3.1 above is the Rastrigin function for PSO, SPSO and APSO respectively. This highly multimodal function has several local minima which are regularly distributed throughout the iteration for all the 3 algorithms. PSO converged to global optimum at the 320th iteration, for SPSO there global optimum was reached at the 192nd iteration, while APSO has its global optimum at

the 57th iteration. The (ab) – (ad) in the above figure is the Rosenbrock function for PSO, SPSO and APSO respectively. For this function also called banana function, PSO have many local minima but convergence to global optimum was reached at the 390th iteration. SPSO have its global optimum at the 270th iteration, while APSO reached its global optimum at the 320th iteration.

The (ae) – (ag) above is the Schaffer f6 function for PSO, SPSO and APSO. This multimodal function is very difficult for most hill-climbing and reactive search algorithm. There was quick convergence for PSO, SPSO and APSO. Global optima for APSO and SPSO were at the 1st iteration, while PSO converged to global optimum at the 10th iteration. The (ah) – (aj) above is the Schaffer f6 modified function for PSO, SPSO and APSO. PSO, SPSO and APSO converged quickly. Global optimum for PSO was at the 4th iteration, for SPSO it was at the 10th iteration and APSO reached the global optimum at the 7th iteration. The (ak) – (am) above is the Tripod function for PSO, SPSO and APSO. Global optimum for PSO was at the 3rd iteration, for SPSO it was at the 6th iteration and APSO reached the global optimum at the 1st iteration. The (an) – (ap) above is the Schaffer f6 Bubble Dynamic function for PSO, SPSO and APSO. The bubbles magnitude cycles up and down. PSO have many local minima as the environment changes. We have global optima for PSO from the 55th to 100th iterations, and from the 200th to 230th and 325th to 352nd iteration. SPSO also have many local minima which are evenly distributed throughout the iteration. We have global optima between 148th and 180th, 270th to 300th and 380th to 400th iterations. APSO also have many evenly distributed local optima in the iteration. Global optima were reached between 45th and 55th, 150th and 180th, and the 400th iterations. The (aq) – (as) in figures above is the Schaffer f6 Linear Dynamic function for PSO, SPSO and APSO. For the 3 PSO algorithms used in this experiment the function moves linearly along a 45 degree angle. PSO has its global optimum at the 150th iteration, SPSO at the 379th iteration and APSO at the

179th iteration. The (at) – (av) in figure 3.1 above is the Schaffer f6 Spiral Dynamic function for PSO, SPSO and APSO. PSO has its global optimum at the 130th iteration, SPSO at the 32nd iteration and APSO at the 48th iteration.

Generally, from our observation, the three variant PSOs experimented were able to converge to their respective local optimal and global optimum. However, there are exceptions where by both the PSO and SPSO failed to find their local and global optimum on the Ackley, Alpine, Dejong f2, Dejong f4, and ND Parabola functions. We observed that the PSO converged 10% faster for the Ackley function while APSO converged 20% faster for the same function as compared to SPSO. APSO has the best performance because the particle's velocity and position are updated immediately after the fitness is computed, thus update occurs with incomplete information about particles in the neighborhood. We suspect that SPSO did not converge slower due to the deep local minima presented by the Ackley function. Similar observation can be made for the Alpine function. Theoretically, Alpine function has many local and global minima with the values of zero and as expected, SPSO could not tackle these different plateaus and was outpaced by PSO (converged 5% faster) and APSO (converged 20% faster).

PSO converged 5% faster as compared to SPSP and APSO for the De Jong f2 function. The performance of the SPSO and APSO are relatively the similar for this function as require longer time to converge. De Jong f2 function is a unimodal function with complicated search landscape comprises of very narrow ridges. For De Jong f4 function, none of the particles reached convergence. This is simply because De Jong f4 is a noisy function. It is therefore likely that the particles have been trapped in the local minima. The three PSO algorithms did not converge under the NDParabola function as well. This is because NDParabola only works efficiently with gradient methods, and therefore possess a challenge for particle swarm algorithms (which is a stochastic method).

Under the Rastrigin function, PSO converged faster 19.75% of the time, SPSO converged faster 52.5% of the time and APSO converged faster 87% of the time. Rastrigin is a multi-modal version of the sphere function with the addition of cosine modulation to produce frequent local minima. It contains relatively millions of local optima. PSO converged faster 72.5% of the time; SPSO converged faster 88% of the time, while APSO converged faster 90% of the time under Griewank. The function Griewank is a highly multimodal function due to the addition of the cosine modulation that produces many widespread local minima. PSO converged faster 60% of the time; SPSO converged faster 89% of the time, while APSO converged faster 88% of the time under Shekel's foxhole, which is a multimodal test function.

PSO, SPSO and APSO all experienced premature convergence very early under the Schaffer f6, Schaffer f6 modified, and Tripod functions. This is because the Schaffer f6 is a complex multimodal function with circular local maxima. Most hill-climbing and reactive search methods find this very difficult to tackle (Roberto, Mauro and Srinivas (2005)). In addition, the Schaffer f6 modified have different centers to look for local minimum, while Tripod is a semi-continuous function, and this presents a problem that makes many algorithms such as PSO and GA to be easily trapped in the local optima (Ashish Raj (2013)).

The F6 Linear Dynamic, F6 Bubbles Dynamic, and F6 Spiral Dynamic all have several local optimal. F6 Linear Dynamic is a version of Schaffer f6 that moves the optima minimum linearly along a 45-degree angle in x, y space. The function F6 Bubbles Dynamic is made up of Schaffer f6 in which each goes on bubbles magnitude cycles up and down which are 180 degree out of phase with each other. Moreover, the F6 Spiral Dynamic is a version of Schaffer f6 that moves the minimum about a Fermat spiral according to the equation: r = a*(theta^2). Where theta is a function of time and is checked internally. x_{centre} = r (cos(theta)) and y_{centre} = r (sin(theta)).

In conclusion, we can deduce from our simulation results that Particle Swarm Optimisation (PSO) still suffers from issues such as premature convergence, inability to effectively cope with dynamic environment and failure of PSO particles to escape from been trapped in local minima. Based on the analysis of our results for the sixteen (16) benchmarking functions used, the presence of premature convergence is confirmed in ten (10) out of the sixteen (16) functions used for PSO, SPSO and APSO. The functions where premature convergence are noticed include Ackley, Dejong f3, Griewank, Foxhole, Schaffer f6, Schaffer f6 modified, Tripod, f6 Bubble Dynamic, f6 Linear Dynamic, and f6 spiral dynamic functions. The inability of PSO and its variants to escape been trapped in the local minima is visible in 9 out of the 16 functions that we used to carry out our tests. PSO was trapped in the local minima in Dejong f2, Dejong f3, Ackley, Alpine, Foxhole, Griewank, Rastrigin, Rosenbrock and Tripod functions. The simulation results from Schaffer f6 Bubble dynamic, Schaffer f6 linear dynamic, and Schaffer f6 spiral dynamic show that PSO, SPSO and APSO does not have the capability to effectively handle optimisation problems in a dynamic environment. The conditions that effect these flaws in the PSO algorithm and that of its variants is that particles in PSO are naturally inclined to fling to the infeasible areas from the feasible areas during the course of searching. This poses a threat to the searching efficiency of PSO.

CHAPTER 4: DEVELOPMENT OF HYBRID ALGORITHMS: PRIMAL-DUAL AND PARTICLE SWARM (*pdPSO*) AND PRIMAL-DUAL AND ASYNCHRONOUS PARTICLE SWARM (*pdAPSO*)

4.1 Introduction

From our literature review in Chapter two, we discovered that the PSO algorithm is an important algorithm for solving various optimisation problems. This is due to the high flexibility and simplicity of implementation of the algorithm. The comprehensive discussion of the applicability of PSO as an optimisation algorithm to support swarm robotics (or multi-objective dynamic optimisation in general) presented in the previous chapter, exposed critical limitations that describes the recent trends of customizing a new swarm algorithms for each swarm robotic project. These limitations can be addressed with the introduction of some other technique to improve the performance of PSO so that it can tackle dynamic optimisation problems adequately. We hypothesize the applicability of embedding an Interior Point Method optimisation (Luke, 2010) to address some of the problem associated with the existing variants of PSO that we have discussed such as premature convergence, and the challenge of some of the particles being trapped in local minima. Why? Evidently, this would propel the initiative towards the creation of a general PSO that can easily be adapted to any swarm robotics project without the need of heavy customization. The Interior-Point algorithm has become recognised as the most ideal approach for solving large-scale linear problems (Laird, 2006). The Primal Dual has been applied to convex optimisation problems where strong duality is required (Rockefeller, 1970). It has also been used for various nonlinear and non-smooth cost functions that are prevalent in network design, medical image reconstruction, and industrial engineering (Boyd & Vandenberghe, 2004). They can be easily

parallelized which enables them to efficiently handle multi-dimensional problems. Primal dual from literature can solve linear optimisation problems effectively (Laird, 2006).

In this chapter, we proposed a hybrid PSO algorithm that will be able to solve the aforementioned problems that are associated with PSO. The Primal Dual algorithm, when integrated into PSO will provide better balance between exploration and exploitation, preventing the particles from experiencing premature convergence and been trapped in local minima easily and so producing better results. The fusion of conventional PSO with Primal-Dual Interior-Point method will resolve the common issues associated with PSO algorithm and many of its variants. The integration will make the system to have great capacity to prevent premature convergence, and prevent the particles from being stuck in local minima. The two key components of this implementation are the explorative capacity of PSO, and the exploitative capability of the Primal-Dual Interior-Point algorithm. On the one hand, exploration is key in searching (i.e., traversing the search landscape) to provide reliable approximation values of the global optimal (Abraham, Pant, Bouvry & Thangaraj, 2011). On the other, exploitation is critical to focus the search on the ideal solutions resulting from exploration to produce more refined results (Torn, 1989). The representation of PSO particle position and velocity update is shown in figure 4.1.

Where

X_i = the position of a particle

V_i = the velocity of the particle

N = the number of particles in the swarm

i = the particle's number (where i = 1…N)

The i^{th} particle is represented as $X_I = (X_{i!}, X_{i2}, ..., X_{iN})$. While the velocity is the rate

at which the next position is changing with respect to the current position. $V_I = (V_{i1}, V_{i2}, ..., V_{iN})$ represent the velocity for the particle i. At the start of the algorithm, initial numerical values of the position and velocity of the particles are assigned haphazardly. This is followed by using equations (1) and (2) to update the position and velocity of the particles after subsequent iterations are conducted during the search process.

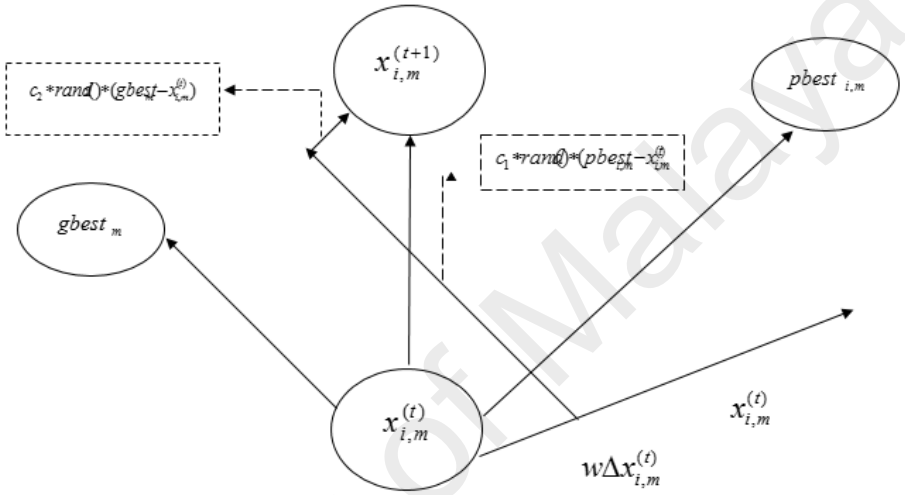


Figure 4.1: Diagrammatic representation of PSO particle position and velocity update

$$v_{i,m}^{(t+1)} = w * v_{i,m}^{(t)} + c_1 * rand_1() * (pbest_{i,m} - x_{i,m}^{(t)} + c_2 * rand_2() * (gbest_m - x_{i,m}^{(t)}) \quad (1)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad (2)$$

In a PSO algorithm, all the particles are randomly introduced and evaluated to calculate fitness of the particles in the swarm. It also computes the local best which is the best value of individual particle and global best which is the best value of particle in the entire swarm (Talukder, 2011). To get the optimum solution, some iterative steps are involved.

During the looping process, the velocity of the particles is first updated by the local and global bests. After this, the position of the individual particle is then updated by the up-to-date velocity of the particle. Once the stopping criteria which has already been predetermined is satisfied, the loop will be terminated.

4.2 Constraint Handling Techniques

One of the limitations of the PSO algorithm is that it is only effective in providing solution to unconstrained global optimisation problems. PSO cannot efficiently solve constrained optimisation problems. There is therefore the need to employ other techniques to handle the constraints. This section will discuss some of the standard approaches that can be used to work out solution to optimisation problems that are constrained in nature.

4.2.1 The Penalty Function Method

In Interior point method, the constrained optimisation problem is solved by changing the problem to a set of unconstrained optimisation problems using the penalty function method. The penalty function is used as a substitute for the objective function. The addition of a penalty parameter to some degree of infringement on the constraints results into the origination of the penalty function. The feasible points and the infeasible points are the two categories of anticipated outcome that guides the search space we have in constrained problems. The measure of degree of infringement on the constraints is zero and non-zero for feasible points and infeasible points respectively.

Stationary penalty function and non-stationary penalty function are the two main classes of penalty functions. The stationary penalty function makes use of penalty values that are constant for the whole operation of minimizing the constrained problem. The non-

stationary penalty function uses dynamic functions that are adjusted in the course of the minimization operation. The outcome of the non-stationary penalty method is usually outstanding when compared with that of the stationary penalty method. To ensure convergence for this class of penalty function, it is required that a very high penalty parameter be used. The side effect of using very high penalty parameter is that it can result into complexity of numbers (Boyd and Vandenberghe, 2004).

4.2.2 Augmented Lagrangian Multiplier Method

The major shortcoming of the Penalty Function method which is the use of large penalty parameter in the penalty function is overcome in the Augmented Lagrangian Multiplier Method. This approach makes use of the combination of Lagrange multiplier and constrained function. The Augmented Lagrangian Multiplier Method is the multiplication of the Lagrange multiplier and constrained function. The constrained optimisation problem just like in the case of penalty function is converted into unconstrained optimisation problem by the Lagrange function. The Lagrangian function can be written as:

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m_g} \lambda_i g_i(x) + \sum_{j=1}^{m_g} \lambda_{j+m_g}\, h_j(x) \qquad (4.1)$$

Where

λ = The Lagrange multiplier

f(x) = The objective function

g(x) = Equality Constraint Violation

h(x) = Inequality Constraint Violation

The solution X^* to constrained problem with the correct set of multipliers λ^* is a stationary point on L. But X^* is not necessarily a minimum of Lagrange function L. To convert the solution X^* from stationary point to minimum, the Lagrange function is augmented using quadratic extension.

$$L_A(x, \lambda, r_p) = f(x) + \sum_{i=1}^{m_e+m_i} \lambda_i \, \theta_i(x) + \sum_{i=1}^{m_e+m_i} r_{p,i} \, \theta_i^2(x) \qquad (4.2)$$

with

$$\theta_i = \begin{bmatrix} g_i(x), & i = 1(1)m_e, \\ \max\left\{ h_{i-m_g(x), \frac{-\lambda_j}{2r_{p,j}}} \right\}, & i = m_e + 1(1)m_e + m_i \end{bmatrix} \qquad (4.3)$$

From the gradient based optimisation problems, the term $\frac{-\lambda_j}{2r_{p,j}}$ in equation 4.3 is selected to have continuous derivatives $\frac{\partial L_A}{\partial X}$ at $\dot{X}$, where $h_{i-me}\dot{X} = \frac{-\lambda_j}{2r_{p,j}}$. The Lagrange function differs from penalty function due to addition of Lagrangian multipliers. In this method, each constraint infeasibility is penalized separately by the use of a vector of positive penalty factors r_p. It can be shown that there exist finite constraint penalty factors r_p in the solution X* of the Lagrange function and thus of the original constrained problem. But the proper values of Lagrange multipliers λ_i and penalty factor parameter r_p are unknown and are always problem dependent and thus the solution of the Lagrange function cannot be computed by single unconstrained minimization of equation 4.2 but a sequence of unconstrained sub-problems with subsequent updates of λ_i and r_p. The update scheme of Lagrange multipliers is based on the solution x^{*v} of the stationary condition of the v-th sub-problem. It holds for $x^v \approx x^{*v}$

$$\left[\frac{\partial f(x)}{\partial x} + \sum_{i=1}^{m_e+m_i} \lambda_i^v \frac{\partial \theta_i(x)}{\partial x} + \sum_{i=1}^{m_e+m_i} 2r_{p,j}^v \, \theta_i(x) + \frac{\partial \theta_i(x)}{\partial x} \right] x = x^v \qquad (4.4)$$

The Lagrange multiplier can be formulated by comparing equations (3.4) and (3.1)

$$\lambda_i^{v+1} = \lambda_i^v + 2r_{p,j}^v \theta_i(x) \tag{4.5}$$

4.2.3 Primal Dual Particle Swarm Optimisation (*pdPSO*)

The Interior-Point algorithm can be used to solve convex problems that have both equality and inequality constraints. Some state of the art of Interior-Point algorithms can be used to handles such problems effectively. The Interior-Point algorithm can be used to compute the solution to the following optimisation problem:

minimize f(x)

subject to c(x) < 0,

where f(x) is a convex objective and c(x) is a vector-valued function with outputs that are convex in x. The input X_0 is the starting point for the solver. It must be an n x 1 matrix, where n is the number of (primal) optimisation variables. The descent direction can either be 'newton' for the Newton search direction, 'BFGS' for the quasi-Newton search direction with the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update, or 'steepest' for the steepest descent direction. The steepest descent direction is often quite bad, and the algorithm may fail to converge to the solution if the option is chosen. For the Newton direction, the Hessian of the objective must be calculated. The quasi-Newton estimate to the objective is usually formed, the implication of this is that a second-order information about the inequality constraint functions. The Interior Point Method algorithm below was adapted from Gilbert, Armand and Jan-Jégou (2000). The algorithm for the Primal Dual Method is below:

Step 1:Initialize some of the algorithm parameters (such as The maximum centering parameter, The maximum forcing number, Minimum barrier parameter, Maximum step size, Minimum step size, Granularity of backtracking search, Amount of actual decrease we will accept in line search).
Step 2: Compute the responses of the unperturbed Karush-Kuhn-Tucker optimality conditions.
Step 3: Check for convergence
Step 4: Update the BFGS approximation to the Hessian of the objective.
Step 5: Find Solution to perturbed KKT system.
Step 7: Do Backtracking line search.
Step 8: Compute the response of the merit function and the directional gradient at the current point and search direction.
Step 9: Compute the candidate point, the constraints, and the response of the objective function and merit function at the candidate point.
Step 10: Stop backtracking search if we've found a candidate point that sufficiently decreases the merit function and satisfies all the constraints.
Step 11: Decrease the step size if candidate point does not meet our criteria
Step 12: Compute the response of the merit function at (x, z).
Step 13: End

Figure 4.2: Primal Dual Method algorithm.

In an attempt to provide solution to the problem of premature convergence and particles being trapped in the local minima that have characterized the PSO algorithm, we hereby propose a hybrid PSO algorithm that can tackle the shortcomings of PSO algorithms and it variants. Primal Dual, when integrated into PSO will provide better balance between exploration and exploitation. The hybridization will enable the system to possess great ability to prevent premature convergence, and prevent the particles from being stuck in the local minimal. The new algorithm works by randomly generating the initial parameters for Primal-Dual and the PSO. The Primal-Dual and PSO operators are applied to generate the initial population. The particles that made up the initial generation are evaluated. When the solution generated is feasible, and then the stopping criteria are checked after which the solution is feed into the Primal-Dual method as the objective function. The result of the Primal-Dual optimisation is then passed to PSO, which creates a perturbation in the population and also

maintain diversity in the population until there is either convergence to the global optimal or the termination criteria is satisfied. The flowchart of Primal Dual PSO (*pdPSO*) is presented in figure 4.3.

The two algorithms that are hybridised in the *pdPSO*: a deterministic algorithm (the Primal-Dual algorithm) and an evolutionary algorithm (the PSO algorithm). The Primal-Dual algorithm possesses a good searching capacity because its rate of convergence is high. It is however liable to converge to a local minima instead of a global optimal solution. In contrast, the PSO algorithm is efficient in converging to a global optimal solution. The implication of this is that a sizable number of particles are required to do a successful search, and consequently large memory capacity and high computation time are needed during the course of optimisation. On the ground of the above rationale, the *pdPSO* algorithm is proposed to solve the problems associated with the PSO algorithm and the Primal-Dual method, and to effectively compute the correct global optimal solution. The flowchart to implement the proposed *pdPSO* algorithm is shown above in figure 4.3.

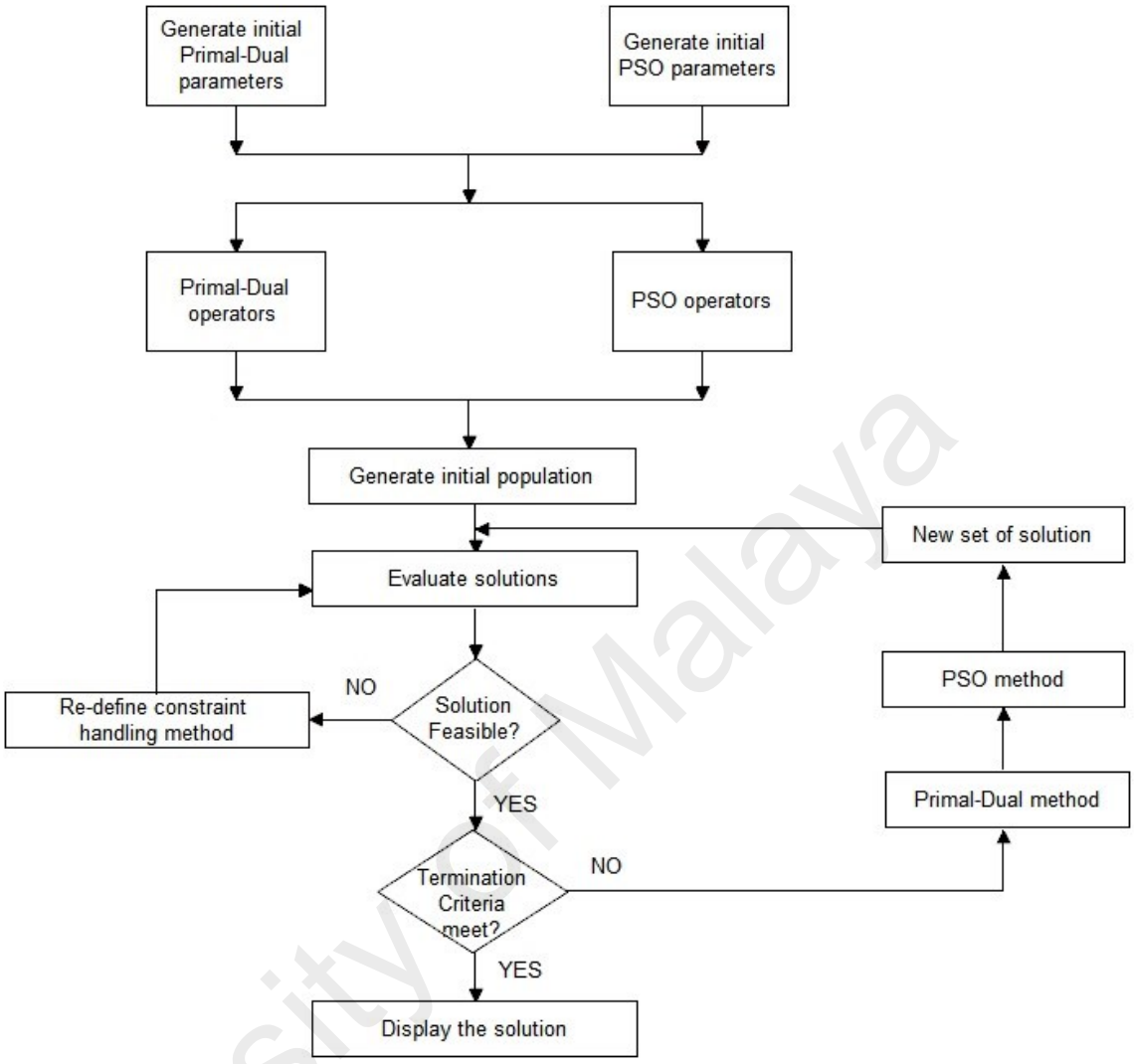


Figure 4**.3:** Flowchart of Primal Dual PSO (*pdPSO*)

4.2.4 Implementation of Primal Dual PSO (*pdPSO*) Algorithm

We implemented our proposed algorithm and carried out some experiments to determine if the algorithm is functioning according to specification. Secondly, we wanted to determine the global optimum and local optima of each of PSO, Primal Dual and *pdPSO* under the different benchmarking functions, and to confirm the problem of the particles in PSO been trapped in the local optimal. Lastly, we are validating the existence of the

premature convergence problem of PSO and Primal Dual Interior Point method.

4.2.4.1 Parameter settings

The dimension value of 30 is assigned for each function (i.e. n = 30). For the implementation of *pdPSO* algorithm mentioned above, a swarm of 50 particles was generated with global best topology. We carried out 400 iterations for each of the algorithm we are testing using the following 9 benchmark functions (Ackley, Sphere, Griewank, Schaffer f6, Schaffer f6 modified, Schaffer f6 Bubble Dynamic, NDParabola, Rastrigin, and Tripod function) running on MATLAB R2012a. The cognitive scaling c_1 that influences local search is set to the value 1.49. Accordingly, the social scaling c_2, which influences the global search, is identically set to the value 1.49. Functions $rand_1$ and $rand_2$ are stochastic variables that have the uniform distribution U (0, 1). Primal-Dual parameter setting is as follows: Primal Dual tolerance is set to 1e-8. The maximum centering parameter is 0.5. The Minimum barrier parameter is set to 1e-9. The Maximum step size is 0.95. The Granularity of backtracking search is set to 0.75

The value of the velocity is limited at $\pm V_{max}$ and the value of V_{max} is set to be equal to the value of X_{max}. This helps in controlling the search range. The range of the searching will become wide if the value assigned is large, thus limiting the algorithm to only global exploration. In contrast, if the value of V_{max} is small the scope of the search will be excessively limited thereby forcing the particles to support only local exploration. The inertia weight *w* called (constriction factor) is the inertia parameter; this regulates the algorithm's searching properties. The initial value is 0.9 and this value decreases to a final value of 0.4. We started with a larger inertia value (a more global search) that is dynamically reduced towards the end of the optimisation (a more local search). Small inertia weight guarantees

quick convergence of the algorithms due to the reduction of time for the exploration in the global space. The inertia weight *w* is used to provide equilibrium between the global and local search capability of the particles in the swarm.

For the primal-dual method, the parameter settings are as follows: Primal-dual tolerance is set to 1e-8. The maximum centering parameter is 0.5. Minimum barrier parameter is 1e-9. Maximum step size is 1e-6, and granularity of backtracking search is 0.75. A total of 20 independent trial runs are carried out for each of the functions that were used to evaluate the performance and robustness of the *pdPSO* algorithm.

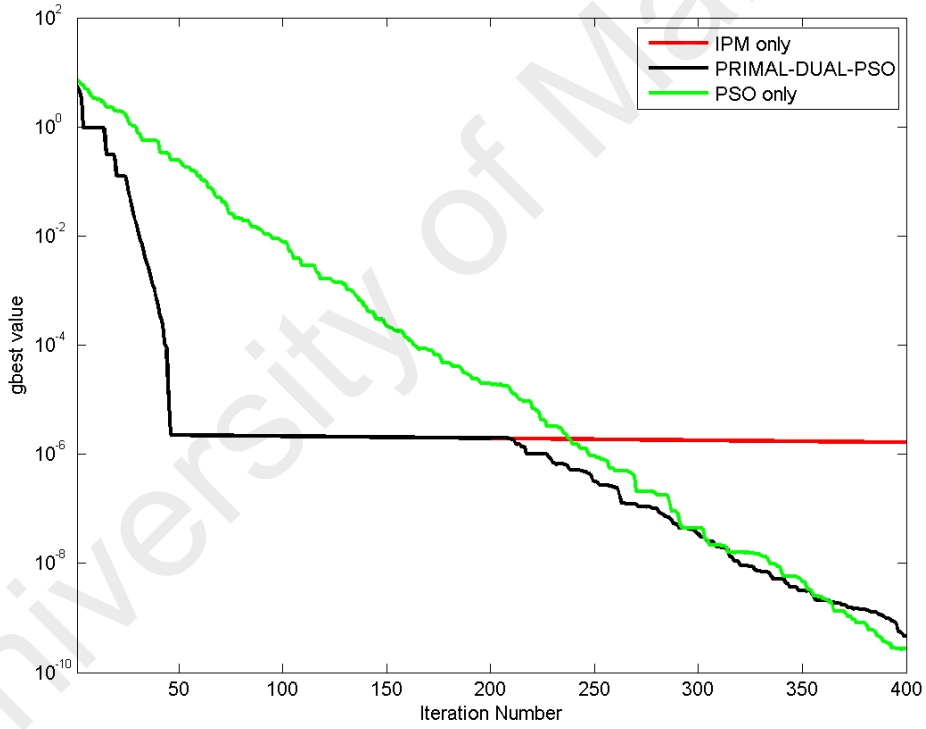


Figure 4.4: Graph of Ackley function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other before IPM got trapped in local minima)

Table 4.1: Result Comparison for Ackley Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	5.79464e-10	2.28608e-05	+9.00249e-07	4.16412e-06
Primal-Dual	1.64406e-06	3.57445e+00	+1.44513e+00	1.35517e+00
PSO	2.72194e-10	2.20607e-08	+1.94965e-09	4.12310e-09

The first function is Ackley (Results depicted in Tab. 4.1). It is a multi-modal function with deep local minima. It has several local minima. It is commonly used to test the ability of the optimisation algorithm to escape local minima. It also used to test the presence of premature convergence in the algorithm. Based on the simulation results for IPM, PSO, and *pdPSO*, there are many local minima generated by the function for PSO and *pdPSO*. PSO and *pdPSO* converged to global optimum, while IPM got trapped in local minima. The convergence rates of PSO and *pdPSO* are almost identical. For the *pdPSO* algorithm, the first 200 iterations were handled by the IPM, while the last 200 iterations were executed by the PSO (thereby combining the exploitative power of IPM and the explorative ability of the PSO). We observed a sharp drop in the *gbest* of *pdPSO* from the first to the 50th iteration, and only after the output of the IPM is inserted into the PSO algorithm the convergence rate doubled. In our comparisons, we used the values of the Best fitness, Mean fitness, and Standard deviation because they are some of the performance measures mentioned in Chena et al. (2010). When we compared the performance of the three algorithms in terms of the numerical values of Best fitness, Mean fitness, and Standard deviation, we can deduce that there is no much significant difference between the performances of PSO and *pdPSO* for Ackley function.

The second function is the Sphere function. It is a unimodal function, it is simple, and there is no communication between the variables. Optimisation algorithms commonly would be to solve the function efficiently. The function can also be used to test the presence of

premature convergence in optimisation algorithms. From the result of our simulation (depicted in Figure. 4.5 and Tab. 4.2), *pdPSO* converged faster than PSO algorithm. For the 250 iterations run, both PSO and *pdPSO* converged successfully. PSO took longer iteration before it converges as it seems to be trapped at a local minimum. This suggests the superiority of the *pdPSO* (in terms of convergence speed) compared to the PSO algorithm. We compared the performance of the three algorithms based on the values of Best fitness, Mean fitness and Standard deviation. From the numerical results, the performance of *pdPSO* was better in terms of the Best fitness, Mean fitness and Standard deviation.

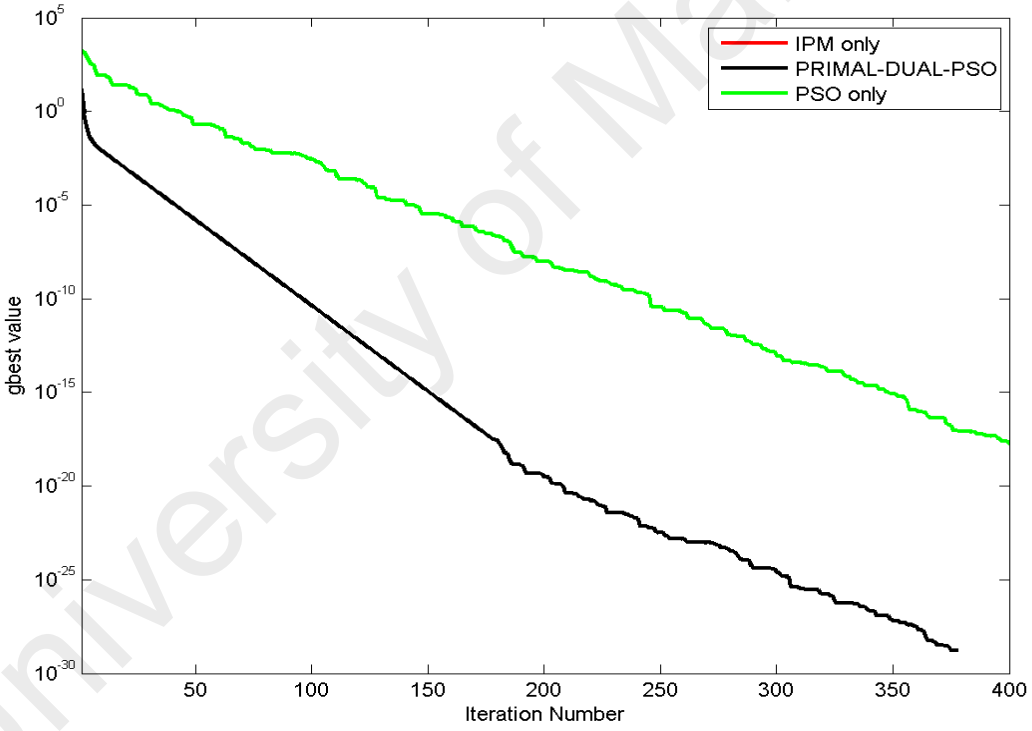


Figure 4.5: Graph of Sphere function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other before)

Table 4.2: Result Comparison for Sphere Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	1.87349e-29	9.11685e-27	+1.19780e-27	2.20741e-27
Primal-Dual	3.05475e-18	1.00235e-17	+6.33670e-18	1.71551e-18
PSO	1.80186e-18	2.76436e-14	+1.05297e-15	5.02818e-15

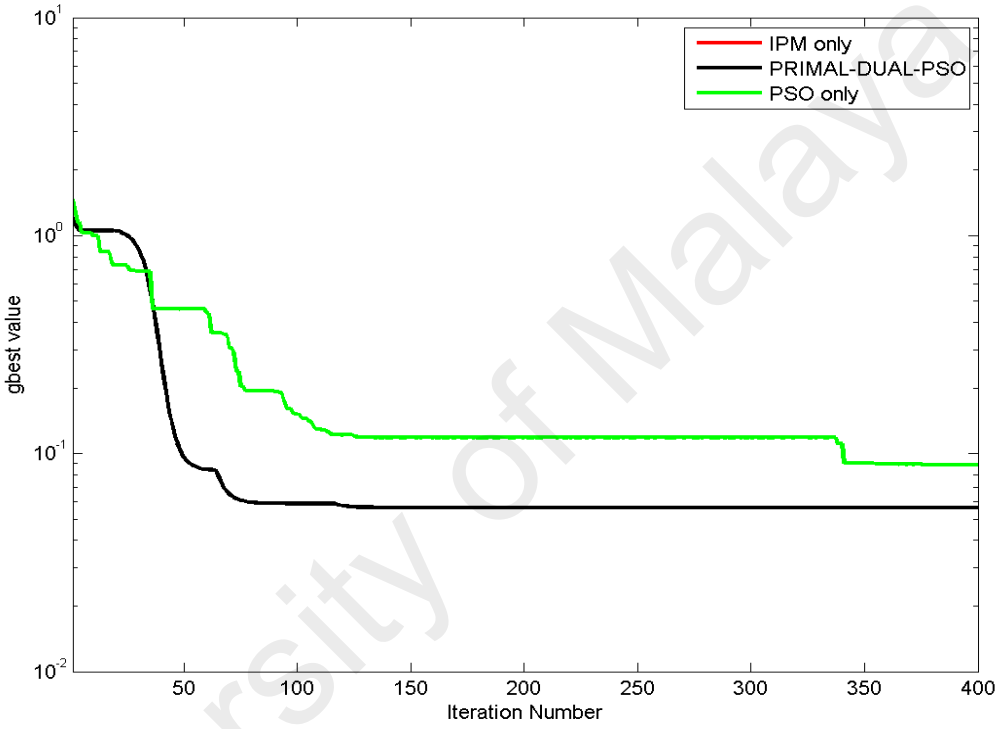


Figure 4.6: Graph of Griewank function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other before they both converged)

Table 4.3: Result Comparison for Griewank Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	5.65695e-02	3.45688e+00	+8.22243e-01	7.07663e-01
Primal-Dual	5.65695e-02	1.08272e+00	+2.73931e-01	3.61745e-01
PSO	8.86239e-02	1.91516e+00	+4.59212e-01	5.75298e-01

The third function is the Griewank function. It is a non-linear multimodal function. It is highly multimodal due to the addition of the cosine modulation that produces many widespread local minima. Griewank has characteristics that are quite similar to that of Rastrigin function except that the number of local optima is more this time around. The numerous local minima have complex structure, and only multi-start algorithms can easily find the global minimum with increase in the dimension of the problem. When used, it tests the ability of the optimisation algorithm to escape been trapped in the local minima. In the Figure above we have the simulation result of Griewank function for IPM, PSO, and *pdPSO*. The PSO has many local optima in the function. All the three (3) algorithms converged to the global optimum for this function. The performance of IPM and *pdPSO* for Griewank function is far better than that of PSO based on their speed of convergence. In term of the numerical value of the Best fitness, the *pdPSO* is superior to the other two algorithms. However, the performance of the Primal-dual is better in terms of Mean fitness and Standard deviation. The *pdPSO* however have a better convergence rate thereby giving it an edge over the Primal-Dual and PSO.

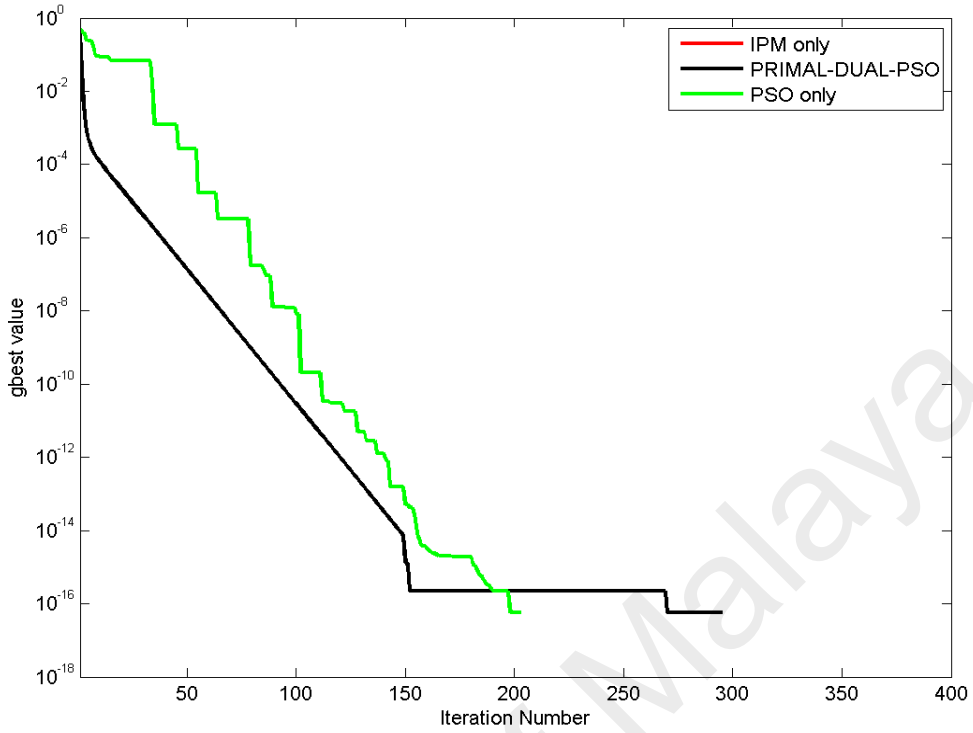


Figure 4.7: Graph of Schaffer f6 function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other).

Table 4.4: Result Comparison for Schaffer f6 Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	0.00000e+00	2.42012e-01	+8.06720e-03	4.41851e-02
Primal-Dual	2.22045e-16	4.24477e-01	+3.33300e-01	1.13677e-01
PSO	0.00000e+00	4.99600e-16	+2.40548e-17	9.29832e-17

The fourth function is the Schaffer f6 Function. It is a complex multimodal function. Most hill-climbing and reactive search methods find it very difficult due to its circular local maxima. It is considered a Genetic Algorithm-hard function to optimize. It is used to test the ability of the optimisation algorithm to escape local minima, as well as premature convergence. The challenge that this function posed to optimisation algorithms is the rise in the magnitude of the prospective maxima, which must be overcome to get to a minimum as

one move nearer to the global minimum. The Schaffer f6 function simulation result for IPM, PSO, and *pdPSO* is depicted in Figure 4.7 and Table 4.4. Apparently, this function poses a challenge to both PSO and *pdPSO* (presence of several local minima). There was a sharp and steady fall in the value of gbest of IPM and *pdPSO* up till the 150th iteration. The *pdPSO* seems to be trapped in a local minimum from the 150th to the 275th iteration after which it experienced a fall in the value of its gbest again. The PSO however converged faster under this function when compared to IPM and *pdPSO*. While *pdPSO* and PSO have better performance in terms of the numerical value of Best fitness, PSO is better in terms of Mean fitness and Standard deviation.

The fifth is Schaffer f6 Modified Function which is the sum of five (5) Schaffer f6 functions with different centres to look for local minimum. It also test the ability of the optimisation algorithm to escape been trapped in the local minima, and check for the presence of premature convergence. The result of the simulation for Schaffer f6 modified function for IPM, PSO, and *pdPSO* is presented in the figure above (depicted in Figure 4.8 and Table 4.5). The IPM and *pdPSO* converged faster than PSO. After the 30th iteration, PSO experienced a sharp fall in the value of its *gbest* and from there got trapped in a local minima and remained there throughout the iteration. There was no significant difference between the performance of the *pdPSO* and the IPM. The *pdPSO* performs better in terms of the value of Best fitness. When we compare the Mean fitness and Standard deviation, the PSO performs better. The *pdPSO* is however superior to the PSO and Primal-dual because it was able to overcome the problem of premature convergence. Whereas the PSO and Primal-dual were trapped in the local minima, *pdPSO* was able to escape been trapped in the local minima.

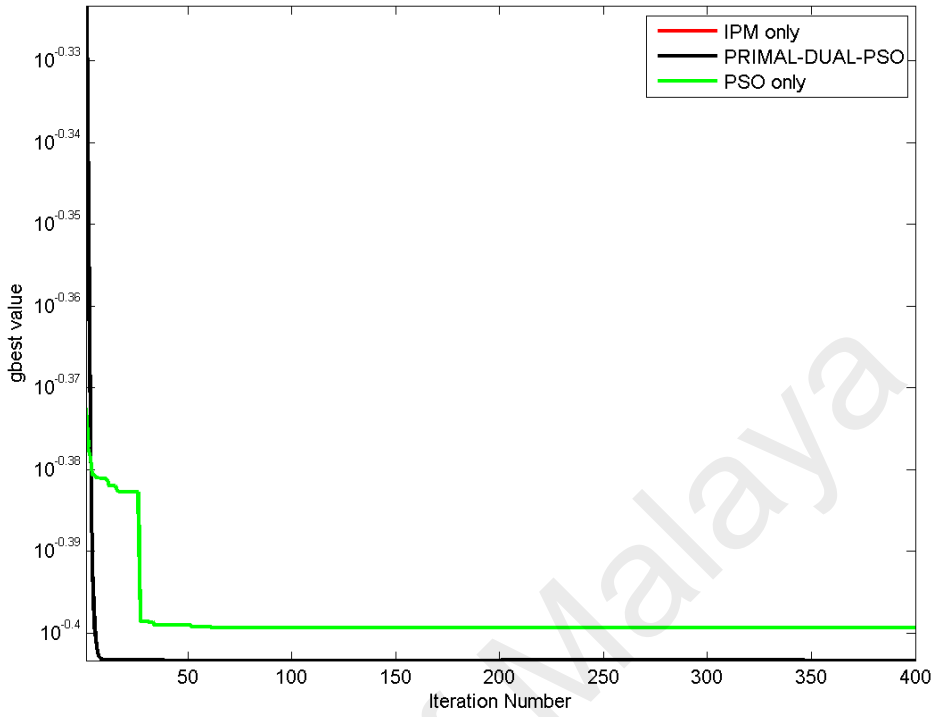


Figure 4.8: Graph of Schaffer f6 Modified function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other before they both converged)

Table 4.5: Result Comparison for Schaffer f6 Modified Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	3.95063e-01	5.66018e-01	+4.03145e-01	3.34147e-02
Primal-Dual	3.95063e-01	4.80710e-01	+4.59061e-01	2.29854e-02
PSO	3.98750e-01	4.84612e-01	+4.01617e-01	1.56753e-02

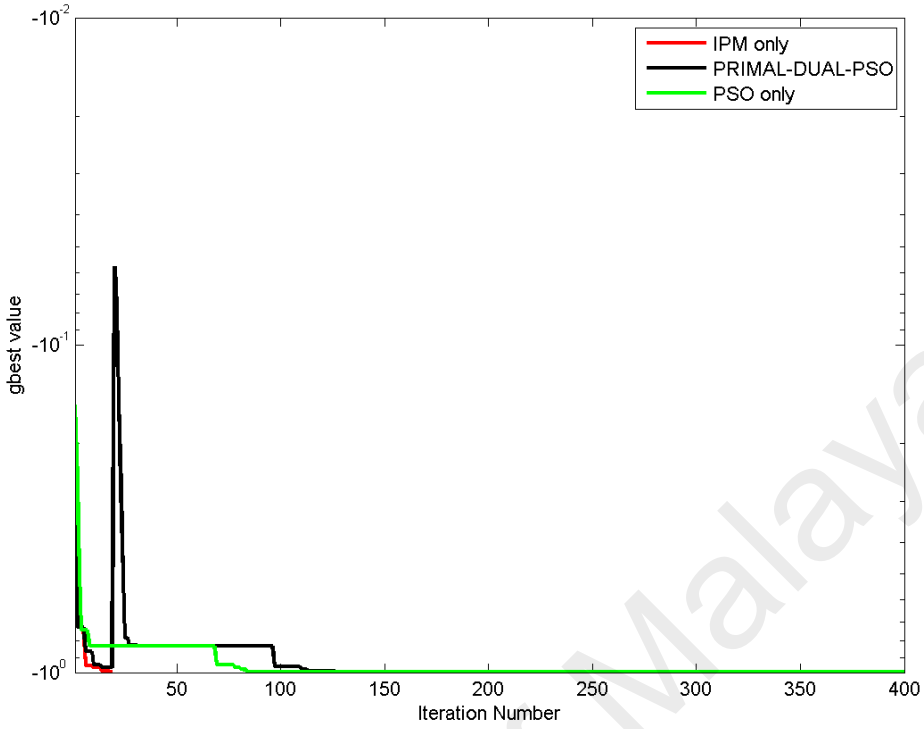


Figure 4.9: Graph of Schaffer f6 Bubble Dynamic function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other)

Table 4.6: Result Comparison for Schaffer f6 Bubble Dynamic Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	5.18555e-02	8.98753e-01	-6.49279e-01	5.05210e-01
Primal-Dual	4.58057e-02	4.43780e-01	-2.89311e-01	9.56216e-02
PSO	9.02080e-01	9.02179e-01	-9.02131e-01	1.50446e-05

The sixth is Schaffer f6 Bubble Dynamic Function which is made up of Schaffer f6 in which each goes on bubbles magnitude cycles up and down. They are 180 degree out of phase with each other. It tests the ability of the algorithm to work effectively in a dynamic environment. The Schaffer f6 Bubble Dynamic function simulation result for IPM, PSO, and *pdPSO* is above (depicted in Figure 4.9 and Table 4.6). The bubbles magnitude cycles up and down especially for IPM and *pdPSO*. Under this function, the *pdPSO* and IPM converged

faster that the PSO. The performance of *pdPSO* is better in terms of the Mean fitness, Primal-dual was better in terms of Best fitness while PSO performs better in terms of Standard deviation. We can deduce that the overall performance of the *pdPSO* is better for this function than that of the other two algorithms we compared because it demonstrate its ability to handle a dynamic environment. This means that *pdPSO* will be more suitable in solving problems that are dynamic in nature compared to standard PSO. While PSO and Primal-dual were static and got trapped in the local minima, *pdPSO* was not.

The seventh function is NDParabola. It was used to test for global minimization problems in Clerc's "semi-continuous challenge." It works very well with gradient methods, but presents a challenge for PSO, which is a stochastic method. The results of our simulation are depicted in Figure 4.10 and Table 4.7. This function tests the ability of the algorithm to converge to global optima after escaping from been trapped in the local minima. The simulation result above is the NDParabola function for IPM, PSO, and *pdPSO*. The IPM, PSO and *pdPSO* algorithms converged to global optima. Both PSO and *pdPSO* have several local minima as shown in the result. There was a great drop in the gbest value of the IPM, PSO, and *pdPSO* from the beginning of the iteration to the end. The convergence speed of *pdPSO* and IPM was much better than that of PSO. Based on the numerical values of the Best fitness, Mean fitness and Standard deviation, *pdPSO* performs better for this function. Our new algorithm (*pdPSO*) also demonstrates its ability to escape been trapped in the local minima and to evade premature convergence in this function.

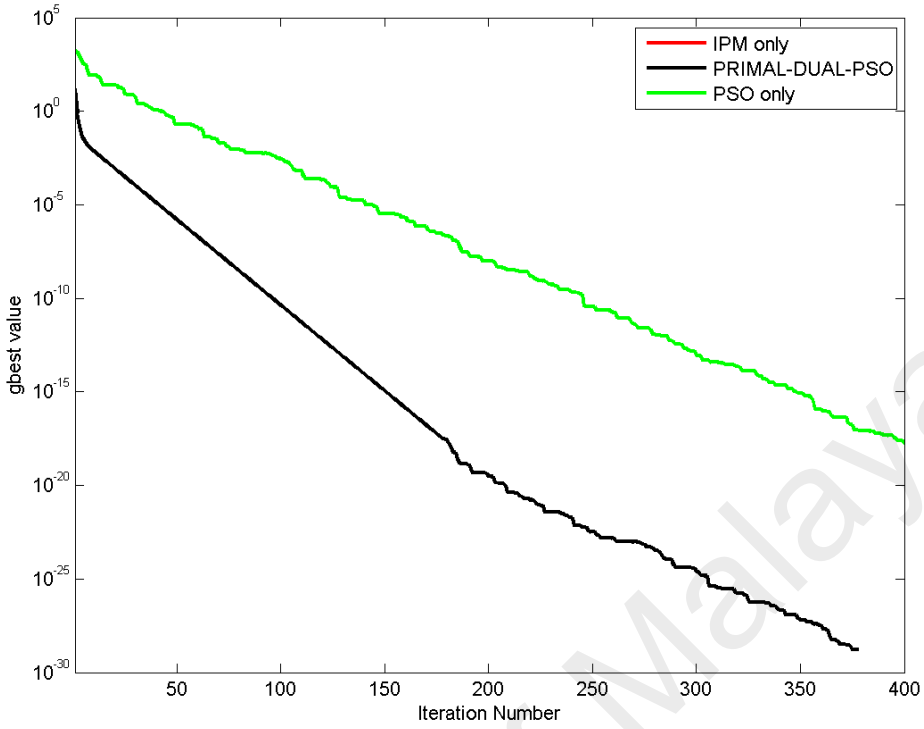


Figure 4.10: Graph of NDParabola function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other)

Table 4.7: Result Comparison for NDParabola Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	1.87351e-29	9.11685e-27	+1.19780e-27	2.20741e-27
Primal-Dual	3.05475e-18	1.00235e-17	+6.33670e-18	1.71551e-18
PSO	1.80186e-18	2.76436e-14	+1.05297e-15	5.02818e-15

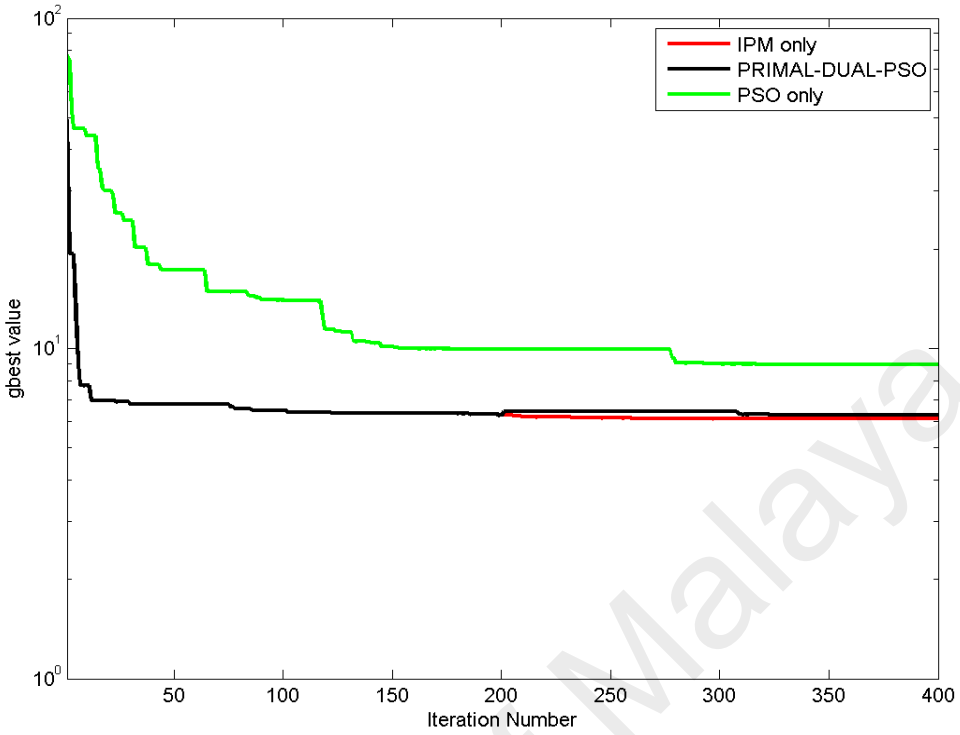


Figure 4.11: Graph of Rastrigin function for Primal-Dual, PSO and *pdPSO*

Table 4.8: Result Comparison for Rastrigin Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	6.29026e+00	1.81783e+02	+7.29321e+01	3.88201e+01
Primal-Dual	6.16813e+00	2.69526e+01	+1.60688e+01	5.13242e+00
PSO	8.95462e+00	8.96635e+00	+8.95527e+00	2.32294e-03

The eighth function is Rastrigin function which is a non-convex, multi-modal version of the sphere function with the addition of cosine modulation to produce frequent local minima. It contains millions of local optima which are organized in a systematic lattice. This function is a moderately problematic one because of its huge search space and its immense number of local minima. The above simulation result is the Rastrigin function for IPM, PSO, and *pdPSO*. This highly multimodal function has several local minima which are regularly distributed throughout the iteration for all the 3 algorithms. The results of our simulation are

depicted in Figure 4.11 and Table 4.8 above. IPM, PSO and *pdPSO* converged to global optimum. The IPM, PSO and *pdPSO* because of the nature of the Rastrigin function have several local minima, and got trapped there for the rest of the iteration. The performance of the IPM and *pdPSO* was a bit better under this function than that of PSO. Judging from the results of the numerical values of the Best fitness and Mean fitness, *pdPSO* performs better. If we consider the value of the Standard deviation, the performance of the PSO seems to be better. The rate of convergence of the *pdPSO* is however superior to that of the other two algorithms. Using the numerical values of the Best fitness and Mean fitness as parameters for our judgment, there was no much difference between the performances of these three algorithms. The performance of *pdPSO* was better based on the numerical value of the Standard deviation when compared to the other two algorithms. From our experiment, *pdPSO* was able to achieve our aim of designing an algorithm that will overcome the problem of premature convergence that usually characterise the standard PSO.

The ninth function is Tripod which is a semi-continuous function. This function presents a problem that many algorithms such as GA and PSO that are easily trapped in one of the two local optima find very difficult to cope with. It is used to test if the optimisation algorithm will be able to escape from been trapped in the local minima, and to also know if it is experiencing premature convergence or not. The Tripod function simulation result for IPM, PSO, and *pdPSO* is depicted in Figure 4.13 and Table 4.9 above. The IPM, PSO and *pdPSO* converged. The *gbest* value of PSO fell sharply from the start of the iteration up till the 10th iteration and decreases steadily from there until it got trapped in a local minimum and remained there throughout the iteration. The IPM and *pdPSO* also experienced fall in the value of their *gbest* from the start up till the 30th iteration after which they got caught in the local minima and remained there till the end of the iteration. Using the numerical values of the Best fitness and Mean fitness as parameters for our judgment, there was no much

difference between the performances of these three algorithms. The performance of *pdPSO* was better based on the numerical value of the Standard deviation when compared to the other two algorithms. From our experiment, *pdPSO* was able to achieve our aim of designing an algorithm that will overcome the problem of premature convergence that usually characterise the standard PSO.

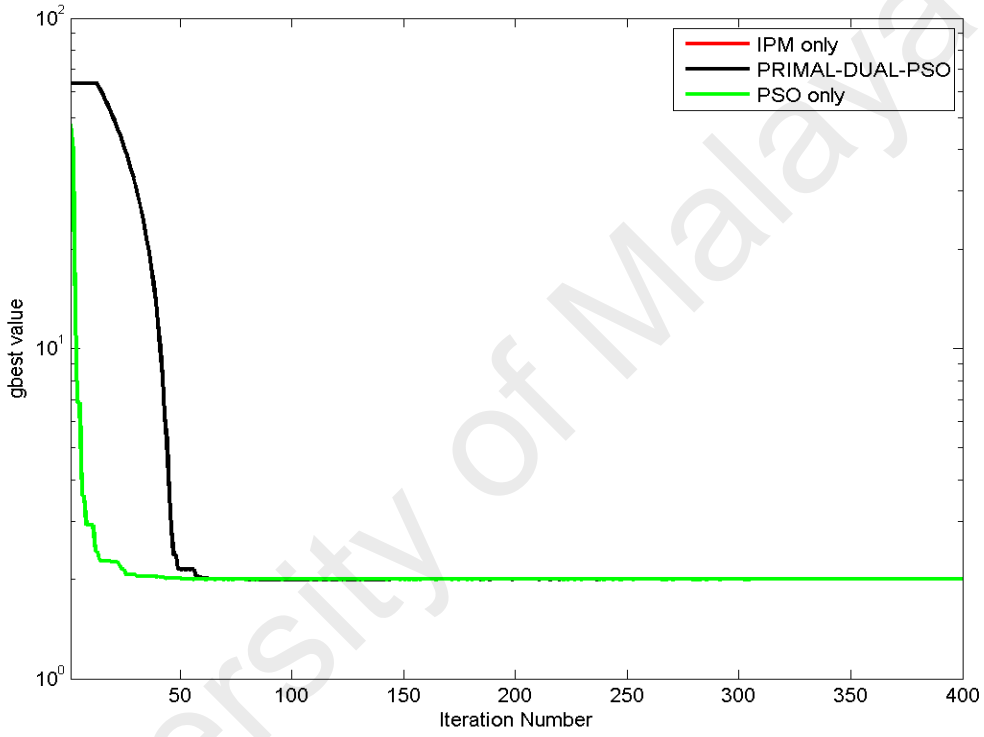


Figure 4.12: Graph of Tripod function for Primal-Dual, PSO and *pdPSO* (IPM and *pdPSO* are superimposed on each other)

Table 4.9: Result Comparison for Tripod Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	2.00000e+00	2.00000e+00	+2.00000e+00	5.94551e-15
Primal-Dual	2.00000e+00	2.00000e+00	+2.00000e+00	1.76109e-08
PSO	2.00000e+00	2.00000e+00	+2.00000e+00	1.96425e-12

Based on the analysis of our results for the nine benchmarking functions used, *pdPSO* performs better than the other algorithms in 7 test cases out of 9; the remaining 2 cases, PSO was superior when compared to other algorithms. The performance of *pdPSO* was superior for Sphere, Griewank, Schaffer f6 modified, Schaffer f6 Bubble dynamic, NDParabola, and Tripod function. The functions where PSO performs better are Ackley and Schaffer f6. The ability of *pdPSO* to overcome the problem of premature convergence and escape been trapped in local minima was demonstrated in the Sphere, Griewank, Schaffer f6 modified, Schaffer f6 Bubble dynamic, and NDParabola. The Tripod function further confirmed the capacity of our algorithm to avoid premature convergence. The results from Schaffer f6 Bubble dynamic shows that *pdPSO* have the capability to handle optimisation problems in a dynamic environment.

With reference to convergence speed, *pdPSO* was faster than PSO and Primal-dual algorithms in 5 functions out of the total of 9 functions that we considered. We can therefore consider *pdPSO* as a fast algorithm that can be used to solve complex numerical optimisation problems. The *pdPSO* also have a higher level of steadiness in comparison to the other two algorithms. The values of Mean fitness and Standard deviations for Sphere, Schaffer f6 Bubble dynamic, NDParabola and Tripod functions were very small when compared to the ones of PSO and Primal-dual. We can there conclude that *pdPSO* is a very steady algorithm that have the capacity to produce rational results that are reliable. Finally we can deduce that *pdPSO* is a robust algorithm as it performs better than PSO and Primal-dual in its ability to successfully find the global optimum on all the benchmarking functions we used especially on Griewank, Schaffer f6, NDParabola and Rastrigin functions which many of the most recent optimisation algorithms finds very problematic to solve. Consequently, *pdPSO* can be considered as an algorithm that can withstand adverse conditions such as the presence of too many local minima.

We went further to conduct more thorough experiments to evaluate the efficiency of the *pdPSO* algorithm. The test functions that we used for the experiments are as shown in the table 4.10 below. Twelve benchmark functions are used in our experiment to further affirm the genuineness of *pdPSO* algorithm. A concise description of these benchmark functions are enumerated in table 4.10 below. Our reason for adopting these benchmark functions is because they have been generally accepted as suitable functions in measuring the performance of global optimisation algorithms (Zhan et al. 2011; Huang et al. 2012; Suganthan et al. 2005; Liang et al. 2005). We made use of twelve functions from the list of functions used in (Gang et al. 2016). Based on the attributes of these functions, they can be categorised into three groups. Category one comprises of three (3) unimodal functions. Category two is made up of four composite multimodal functions. The category three comprises of six functions out of which four are rotated multimodal while the remaining two are shifted functions.

Table 4.10: Test functions used in the comparisons

Function Name	**Dimension (D)**	**Global opt**	**Search Range**	**Initialization Range**
Unimodal				
Sphere	30	$\{0\}^D$	$[-100,100]^D$	$[-100,50]^D$
Schwefel's P2.22	30	$\{0\}^D$	$[-10,10]^D$	$[-10,10]^D$
Rosenbrock	30	$\{0\}^D$	$[-10,10]^D$	$[-10,10]^D$
Multimodal				
Rastrigin	30	$\{0\}^D$	$[-5.12,5.12]^D$	$[-5.12,5.12]^D$
Ackley	30	$\{0\}^D$	$[-32.768,32.768]^D$	$[-32.768,32.768]^D$
Schwefel	30	$\{0\}^D$	$[-500,500]^D$	[-500,500]D
Griewank	30	$\{0\}^D$	$[-600,600]^D$	$[-600,600]^D$
Rotated and Shifted				
Rotated Rosenbrock	30	$\{0\}^D$	$[-10, 10]^D$	$[-10, 10]^D$
Rotated Rastrigin	30	$\{0\}^D$	$[-5.12,5.12]^D$	$[-5.12,5.12]^D$
Rotated Ackley	30	$\{0\}^D$	$[-32.768,32.768]^D$	$[-32.768,32.768]^D$
Rotated Griewank	30	$\{0\}^D$	$[-600,600]^D$	$[-600,600]^D$
Shifted Rosenbrock	30	$\{0\}^D$	$[-100,100]^D$	$[-100,100]^D$

Shifted Rastrigin	30	$\{0\}^D$	$[-100,100]^D$	$[-100,100]^D$

Some performance metrics were used to evaluate the performance of the *pdPSO* in order to know the dependability of the algorithm and the quality of the solution generated. Such performance metrics include the value of mean fitness and standard deviation. The speed of convergence is measured by computing the average number of FEs needed to arrive at a satisfactory solution among successful runs. The dependability of the algorithm is evaluated based on the mean success rate (SR %). The computation of the Mean value of FEs is done only for the successful runs. The ratio of trial runs expressed as a fraction of 100 that successfully reach the standard accuracy is called the success rate. According to Auger and Hansen (2005), some algorithms may fail to attain the satisfactory solution for each run on some problems. Another standard of measurement is called success performance (SP).

Where

Success = (Fit(x*) + (1.0E-5))

X* = Theoretical global optimal solution

NFE = Average number of function evaluation required to find solution when all 30 runs are successful.

SP = (Mean FEs)/(SR%)

The statistical results of the experiments conducted using twelve (12) benchmarking functions for *pdPSO* algorithm is summarized in the table below.

Table 4.11: Statistical result of 12 benchmarking functions for *pdPSO*

Algorithm Name	**Primal-Dual-PSO**							
Function Name	Best Fitness	Worst Fitness	Mean Fitness	Standard Deviation	SP	Success Rate (%)	Runtime (s)	NFE
Sphere	-4.500e+002	3.521e+005	-4.500e+002	6.387e-014	1078	100	177.66	1078
Schwefel's P2.22	-4.500e+002	1.331e+007	-4.500e+002	3.683e-006	3268.36	91.7	142.441	2996
Rosenbrock	3.900e+002	5.407e+011	3.925e+002	5.083e+000	8934.35	80.6	155.42	2997
Rastrigin	-3.300e+002	7.525e+002	-3.300e+002	4.198e-014	2827.00	100	156.97	2827
Ackley	3.850e+002	5.378e+011	3.855e+002	5.052e+000	8821.14	85.7	150.12	8285
Schwefel	-4.495e+002	2.94e+007	-4.437e+002	4.07e+000	5615.45	58.63	124.66	2981
Griewank	-1.800e+002	1.236e+004	-1.800e+002	1.084e-002	6175.20	41.66	186.46	2573
Rotated Rosenbrock	3.88e+002	5.571e+011	3.996e+002	5.542e+000	8986.74	38.79	168.42	3456
Rotated Rastrigin	-1.800e+002	1.236e+004	-1.800e+002	1.084e-002	6175.20	41.66	186.46	2573
Rotated Ackley	-1.39e+002	5.000e+000	-1.392e+002	5.514e-002	3748.29	72.19	189.16	2455
Rotated Griewank	-1.80e+002	2.80e+002	-1.824e+00	1.063e-00	1734.92	68.8	253.23	1124
Shifted Rosenbrock	-3.300e+002	7.519e+002	-3.300e+002	4.198e-014	2827.00	100	156.97	2827
Shifted Rastrigin	3.09e-002	1.899e+003	-2.97e+002	7.180e+000	5873.29	59.83	174.37	2999

4.3 Primal Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*) algorithm

In our effort to improve on the performance of *pdPSO* algorithm, we proposed another algorithm called Primal-Dual APSO (*pdAPSO*). Unlike what we have in the standard PSO, the Asynchronous PSO (APSO), after evaluating the fitness of the swarm, the velocity and position of particles are updated immediately after computing their fitness using partial or limited information about the neighbourhood. This results into varieties in the swarm since some of the information is from the previous iteration while some is from the current iteration. From literature, it has been discovered that APSO performs better and converges faster than standard PSO and SPSO (Luo & Zhang, 2006). Some other researchers concluded from their work that APSO provides the best accuracy at the expense of computational time. In this chapter, we proposed a fusion of Asynchronous PSO with Primal-Dual Interior-Point method to overcome some of the drawbacks of the PSO algorithm. The algorithm for the asynchronous PSO is in figure 4.14.

4.4 Implementation of Primal-Dual Asynchronous PSO (Primal-Dual-APSO)

The purpose of this experiment is firstly to implement our new algorithm called Primal-Dual Asynchronous PSO (Primal-Dual-APSO) and to determine if it is working properly. Secondly, it is to compare the performance of APSO, Primal-Dual-PSO (*pdPSO*) proposed by Dada and Ramlan (2015), and PSO under different benchmarking functions, and to confirm the problem of the particles in PSO been trapped in the local optimal. Thirdly, our purpose is to know the global optimum and local optimal of each of APSO, Primal-Dual-APSO (*pdAPSO*), and PSO under different benchmarking functions. Fourthly, we are authenticating the reality of the premature convergence problem of PSO algorithms. It is obvious from our review of related literature that based on some of the weaknesses of PSO, it is highly essential to develop new variants of PSO algorithms in to order to be able to apply it to solving problems in other fields and thus contribute towards the ever-expanding pool of PSO algorithms. Obviously, the algorithm is not suitable for solving dynamic problems. We hereby present a hybridized APSO and Primal Dual algorithm which is an improvement on the earlier variants that have been developed, thereby contributing to the field of swarm intelligence.

Step 1: Get initial points boundary values

Step 2: Get initial values for particles position
- Step 2.1: Particles best positions = particles positions
- Step 2.2: Particles velocity = particles dimension
- Step 2.3: Get particles best value
- Step 2.4: Best val hist = zeros
- Step 2.5: particles positions = random particles dimension

Step 3: Main loop starts here
- Step 3.1: For iteration = 1 to iteration number
- Step 3.2: Loop through every single particle
- Step 3.3: for temp = 1: particles

Step 4: Update the position of an individual particle
- Step 4.1: Particles positions = particles positions + particles velocity

Step 6: Evaluate the objective function for each particle
- Step 6.1: If objective function < particles best value

Step 7: Update position, (Best position).
- Step 7.1: Particles best positions = particles positions
- Step 7.2: Update best value so far (Particles best value = objective function)
- Step 7.3: End

Step 8: Best function value calculation
- Step 8.1: Compute Gbest val = min (particles best value);
- Step 8.1: Compute Best val hist = gbest;

Step 9: Velocity component update for an individual particle
- Step 9.1: Compute particles velocity = c_1 * particles velocity + c_2 * rand (particles dim) *particles best positions – particles positions + c_2*rand (particles dim) * (particles best positions) – particles positions
- Step 9.1: End

Step 10: End

Figure 4.13: Algorithm for Asynchronous PSO (APSO)

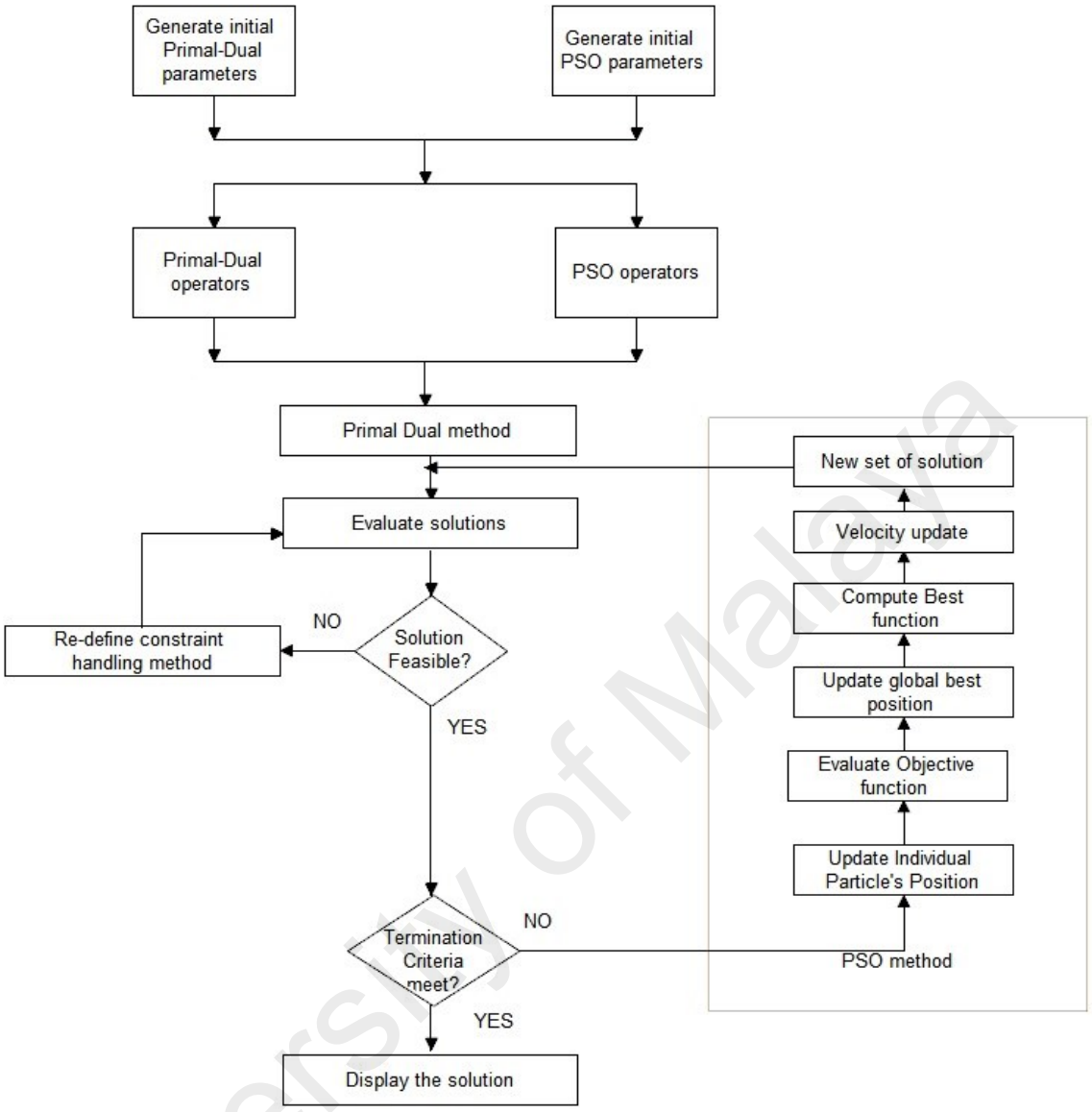


Figure 4.14: Flowchart for *pdAPSO* Algorithm

4.4.1 Parameter settings

The dimension value of 10 is assigned for each function (i.e. n = 10). The size of the swarm is 30. We carried out 400 iterations for each of the algorithm we are testing using the following 9 benchmark functions (Ackley, Dejong f2, Sphere, Griewank, Schaffer f6, Schaffer f6 modified, NDParabola, Rastrigin, and Tripod function) running on MATLAB

R2012a. All the parameters used for the implementation of *pdPSO* were also adopted for *pdAPSO*.

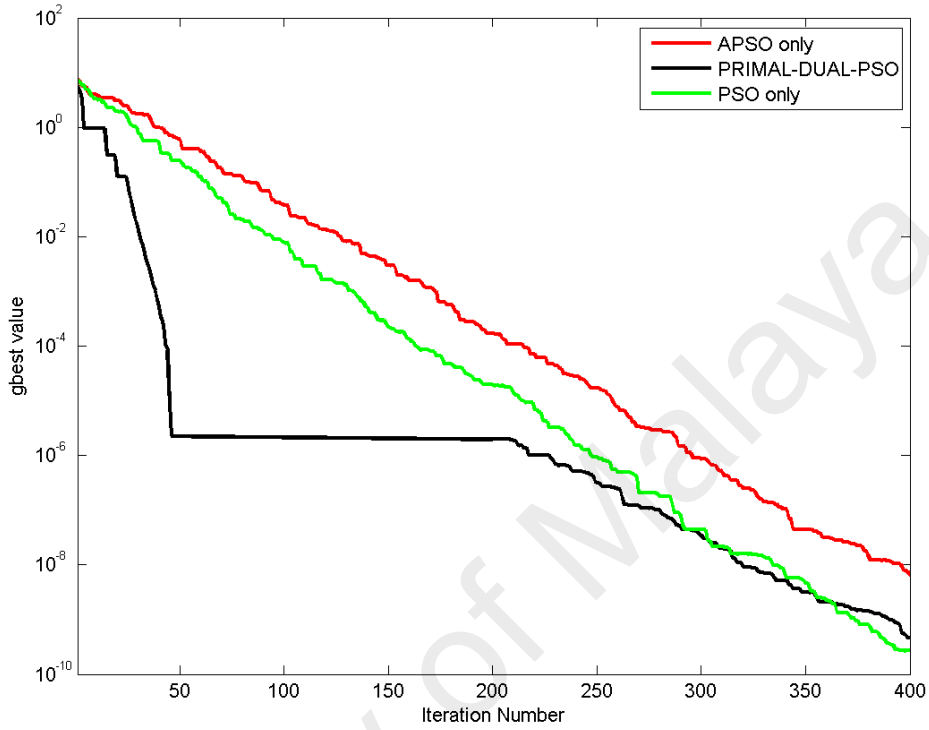


Figure 4.15: Graph of Ackley function for APSO, Primal-Dual-APSO and PSO

Table 4.12: Result Comparison for Ackley Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	5.79464e-10	2.28608e-05	+9.00249e-07	4.16412e-06
APSO	6.51690e-09	7.56732e-08	+1.75608e-08	1.62062e-08
PSO	2.72194e-10	2.20607e-08	+1.94965e-09	4.12310e-09

The simulation results of the APSO, *pdPSO*, and PSO algorithms are provided in the figure 4.16 and table 4.12. The function Ackley is a multi-modal function with deep local minima. It has moderate complexities. Algorithms that only use the gradient steepest descent are likely to be trapped in the local optima under this function. It has several local minima. It

is used to test the ability of the optimisation algorithm to escape been trapped in the local minima. It also tests the presence of premature convergence in these algorithms. Based on the simulation results for APSO, *pdAPSO*, and PSO, there are many local minima generated by the function for PSO and *pdAPSO*. All the three algorithms converged to global optimum. The Primal-Dual-PSO experienced a sharp fall in the value of its gbest from the start of the iteration up till the 50th iteration where the particles stayed on the same gbest till the 200th iteration. The convergence rates of PSO and *pdAPSO* are almost the same. For the *pdAPSO* algorithm, the first 200 iteration was handled by the IPM, while the last 200 iterations were done by the PSO thereby combining the exploitative power of IPM and the explorative ability of the PSO. The performance of APSO under Ackley function is not as good as that of *pdAPSO* and PSO.

The simulation result (depicted as Figure 4.17 and Table 4.13) of the Dejong f2 function. This is a two dimensional function with a subterranean vale that looks like that of a parabola which proceeds to the global minimum. As a result of the non-linearity of the vale, many algorithms find it very difficult to converge quickly under this function because the value changes the direction of the search repeatedly. This function poses a challenge to many optimisation algorithms. All the functions converged, and there are many local minima that are experienced by the APSO, *pdPSO* and PSO. The performance of the APSO was superior (in terms of convergence speed) to that of the PSO and *pdPSO* under this benchmark function. The *pdPSO* was trapped in the local minima from the 10th iteration to the 200th iteration until the result of the optimisation from IPM was fed into PSO.

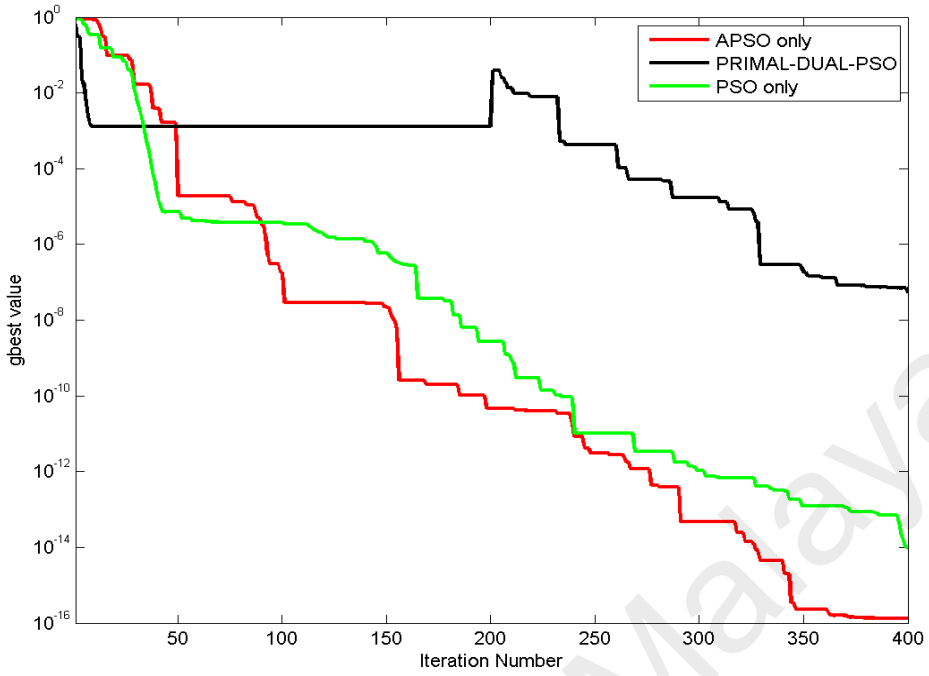


Figure 4.16: Graph of Dejong f2 function for APSO, Primal-Dual-APSO and PSO

Table 4.13: Result Comparison for Dejong f2 Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	5.42346e-08	5.08558e+02	+1.75616e+01	9.27722e+01
APSO	1.34362e-16	5.07726e-03	+1.70015e-04	9.26840e-04
PSO	9.67875e-15	1.42771e+02	+4.75906e+00	2.60663e+01

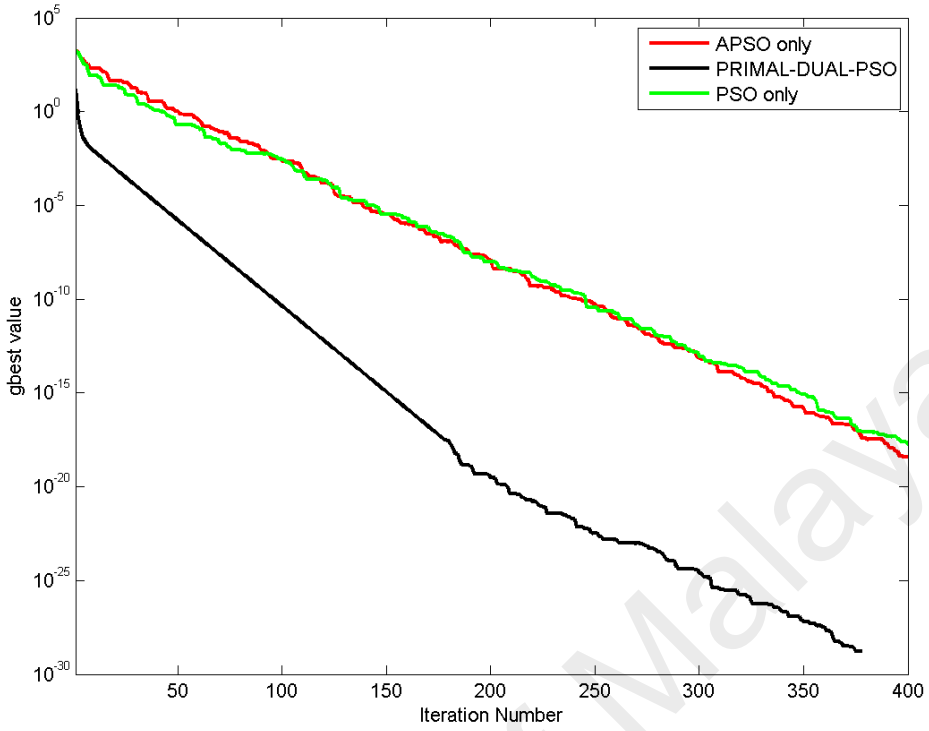


Figure 4.17: Graph of Sphere function for APSO, Primal-Dual-APSO and PSO

Table 4.14: Result Comparison for Sphere Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	1.87351e-29	9.11685e-27	+1.19781e-27	2.20741e-27
APSO	4.81143e-19	9.88981e-16	+9.65854e-17	2.48148e-16
PSO	1.80186e-18	2.76436e-14	+1.05297e-15	5.02818e-15

The above function is the Sphere function. It is a unimodal function, it is simple, and there is no communication between its variables. Our simulation results are depicted in Figure 4.18 and Table 4.14 above. Optimisation algorithms usually do not find it difficulty. It can also be used to test the presence of premature convergence in optimisation algorithms. From the result of our simulation, *pdAPSO* converged faster than APSO and PSO algorithms. In short, the *pdAPSO* is performing better than PSO and APSO. This suggests the superiority

of the *pdAPSO* (in terms of convergence speed) compared to the other two algorithms. The performance of APSO is slightly better than that of the PSO under this function.

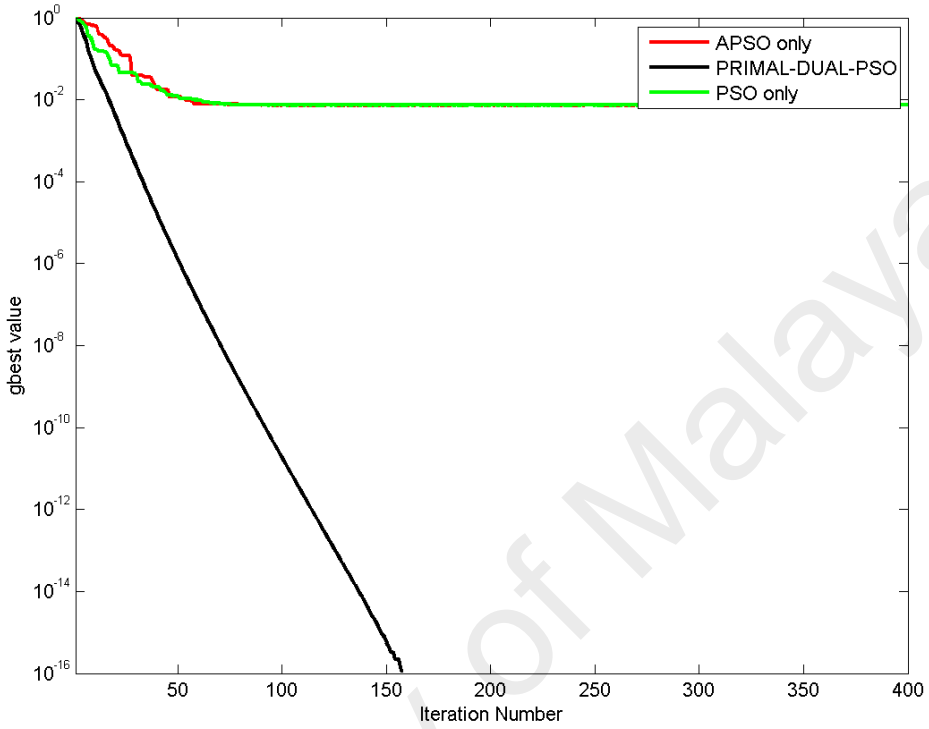


Figure 4.18: Graph of Griewank function for APSO, Primal-Dual-APSO and PSO

Table 4.15: Result Comparison for Griewank Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	0.00000e+00	1.11022e-16	+3.70074e-18	2.02698e-17
APSO	7.39604e-03	7.39604e-03	+7.39604e-03	9.25946e-16
PSO	7.39604e-03	7.39604e-03	+7.39604e-03	1.91040e-16

The simulation result of the Griewank function is above. Griewank is a non-linear multimodal function. It is highly multimodal due to the addition of the cosine modulation that produces many widespread local minima. The optima are regularly distributed. The function was designed to produce interdependence among the variables. Griewank has

characteristics that are quite similar to that that of Rastrigin function except that the number of local optima is more this time around. The numerous local minima have complex structure, and only multi-start algorithms can easily find the global minimum with increase in the dimension of the problem. When used, it tests the ability of the optimisation algorithm to escape been trapped in the local minima. In the figure 4.19 and table 4.15 above we have the simulation result of Griewank function for APSO, *pdAPSO* and PSO. The APSO and PSO experienced many local optima in the function. All the three (3) algorithms converged to the global optimum for this function. The gbest of *pdAPSO* experienced a very sharp fall throughout the iteration. The performance of PSO and APSO is the same under the Griewank function, as both of them got trapped at a point and stayed there till the end of the iteration.

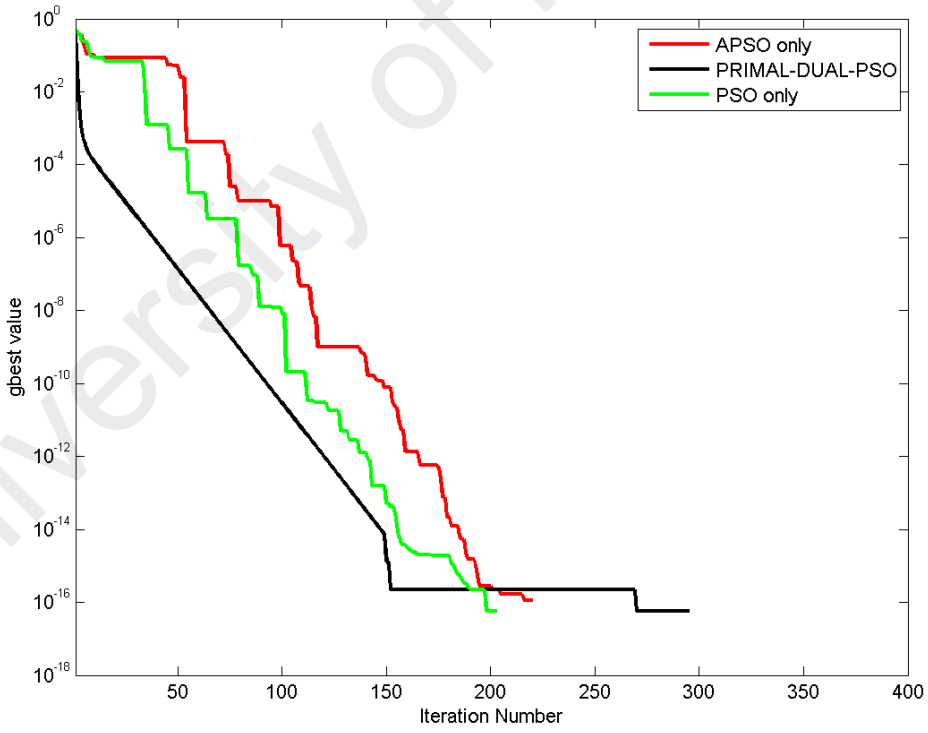


Figure 4.19: Graph of Schaffer f6 function for APSO, Primal-Dual-APSO and PSO

Table 4.16: Result Comparison for Schaffer f6 Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	0.00000e+00	2.42012e-01	+8.06720e-03	4.41851e-02
APSO	0.00000e+00	1.15650e-05	+3.85499e-07	2.11147e-06
PSO	0.00000e+00	4.99600e-16	+2.40548e-17	9.29832e-17

Above in Figure 4.20 and Table 4.16 are the simulations result for APSO, *pdAPSO* and PSO using the Schaffer f6 function. It is a complex multimodal function. Most hill-climbing and reactive search methods find it very difficult due to its circular local maxima. Stochastic algorithms such as PSO and its variants also find it very difficult to cope with it. It is considered a GA-hard function to optimize. Schaffer f6 Function is used to test the ability of the optimisation algorithm to escape been trapped in the local minima, and also test for premature convergence. The challenge that this function posed to optimisation algorithms is the rise in the magnitude of the prospective maxima which must be surmounted to get to a minimum as one move nearer to the global minimum. Both PSO and *pdAPSO* have several local minima. There was a sharp and steady fall in the value of gbest of *pdAPSO* up till the 150th iteration. The *pdAPSO* seems to be trapped in a local minimum from the 150th to the 275th iteration after which it experienced a fall in the value of its gbest again. The PSO however converged faster under this function when compared to APSO and *pdAPSO*. This is because for the *pdAPSO,* after the IPM's 150 iterations, PSO takes over from there.

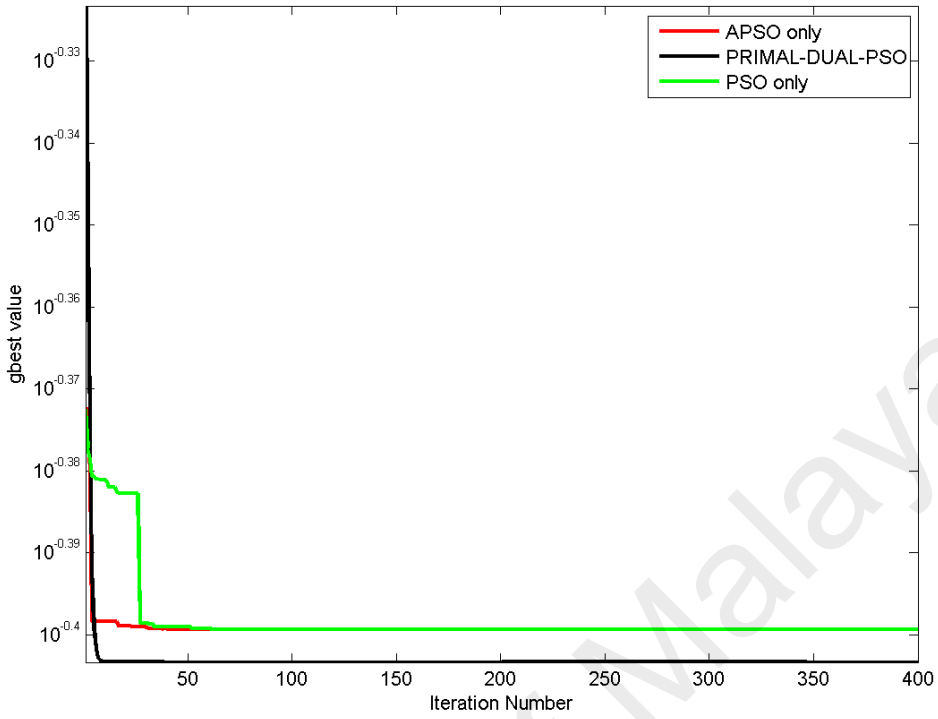


Figure 4.20: Graph of Schaffer f6 Modified function for APSO, Primal-Dual-APSO and PSO

Table 4.17: Result Comparison for Schaffer f6 Modified Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	3.95063e-01	5.66018e-01	+4.03145e-01	3.34147e-02
APSO	3.98750e-01	3.98750e-01	+3.98750e-01	1.55991e-16
PSO	3.98750e-01	4.84612e-01	+4.01617e-01	1.56753e-02

The Schaffer f6 Modified Function is the sum of five (5) Schaffer f6 functions with different centres to look for local minimum. It is used to test the ability of the optimisation algorithm to escape been trapped in the local minima, and check for the presence of premature convergence. The result of the simulation for Schaffer f6 modified function for APSO, PSO, and *pdAPSO* is presented in the figure 4.21 and table 4.17 above. The *pdAPSO* converged

faster than PSO and APSO, this further show the superiority in the performance of our new algorithm when compare to PSO and APSO. There was a sharp fall in the value of the gbest of *pdAPSO* and APSO from the start of the iteration. After the 30th iteration, PSO also experienced a fall in the value of its *gbest* and from there got trapped in local minima and remained there throughout the iteration. PSO converged faster than the APSO.

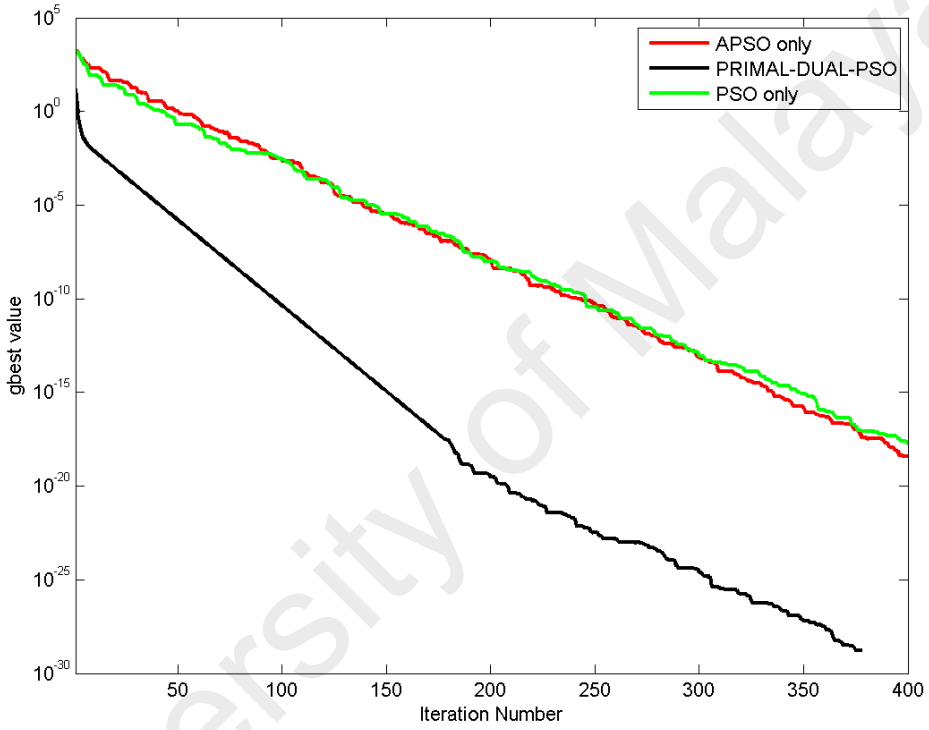


Figure 4.21: Graph of NDParabola function for APSO, Primal-Dual-APSO and PSO

Table 4.18: Result Comparison for NDParabola Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	1.87351e-29	9.11685e-27	+1.19780e-27	2.20741e-27
APSO	4.81143e-19	9.88981e-16	+9.65854e-17	2.48148e-16
PSO	1.80186e-18	2.76436e-14	+1.05297e-15	5.02818e-15

The NDParabola is a benchmarking function that is used to test for global minimization problems in Clerc's "semi-continuous challenge." It works very well with gradient methods, but presents a challenge for PSO, which is a stochastic method. This function tests the ability of the algorithm to converge to global optima after escaping from been trapped in the local minima. The simulation result presented in figure 4.22 and table 4.18 above is the NDParabola function for APSO, PSO, and *pdAPSO*. All the three algorithms converged to global optima, and they all have several local with massive drop in the gbest values from the beginning of the iteration to the end as shown in the result. The convergence speed of *pdAPSO* was much better than that of PSO and APSO.

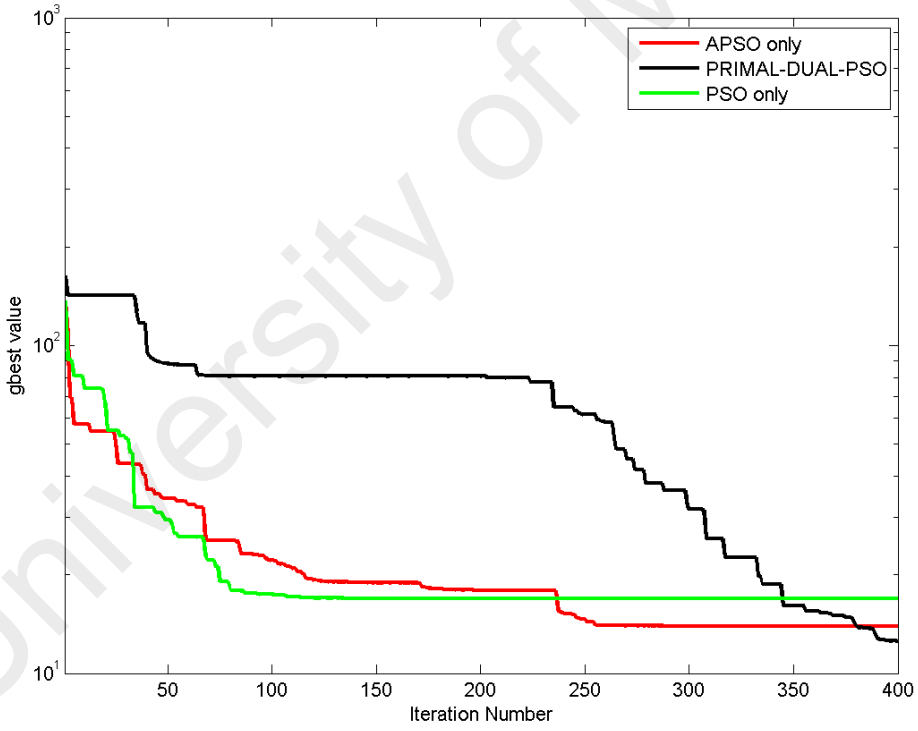


Figure 4.22: Graph of Rastrigin function for APSO, Primal-Dual-APSO and PSO

Table 4.19: Result Comparison for Rastrigin Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	1.25268e+01	2.04591e+02	+4.71626e+01	4.31244e+01
APSO	1.39294e+01	1.39294e+01	+1.39294e+01	7.43627e-06
PSO	1.69143e+01	1.69143e+01	+1.69143e+01	7.28970e-11

The above simulation result depicted in figure 4.23 and table 4.19 is the Rastrigin function for APSO, PSO, and *pdAPSO*. The Rastrigin function is a non-convex, multi-modal version of the sphere function with the addition of cosine modulation to produce frequent local minima. It contains millions of local optima which are organized in a systematic lattice. This function is a moderately problematic one because of its huge search space and its immense number of local minima. This highly multimodal function has several local minima which are regularly distributed throughout the iteration for all the 3 algorithms. The entire algorithm converged to global optima. The PSO because of the nature of the Rastrigin function with several local minima got trapped and stayed there for the rest of the iteration. The performance of the *pdAPSO* was superior to that of the other two algorithms under this function.

The Tripod is a semi-continuous benchmarking function. This function presents a problem that many algorithms such as GA and PSO that are easily trapped in one of the two local optima find very difficult to cope with. It is used to test if the optimisation algorithm will be able to escape from been trapped in the local minima, and to also know if it is experiencing premature convergence or not.

The Tripod function simulation result for APSO, PSO, and *pdAPSO* is in the figure 4.24 and table 4.20. The entire algorithm converged; the APSO and PSO converged faster than *pdAPSO*. All the algorithms experienced fall in the value of their *gbest* from the start up

till the 30th iteration after which they got caught in the local minima and remained there till the end of the iteration.

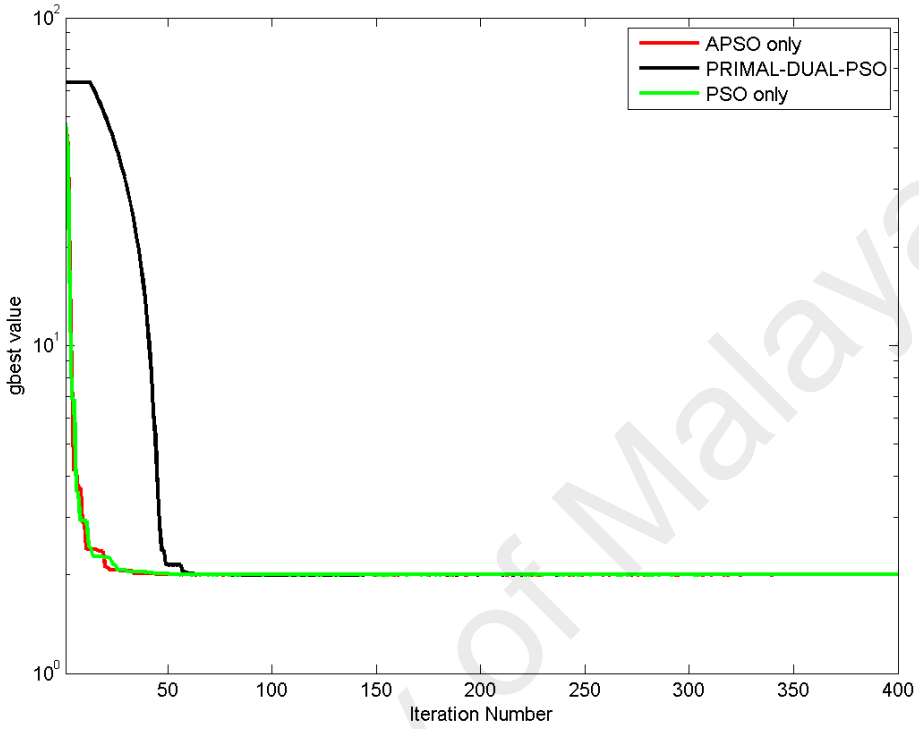


Figure 4.23: Graph of Tripod function for APSO, Primal-Dual-APSO and PSO

Table 4.20: Result Comparison for Tripod Function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-APSO	2.00000e+00	2.00000e+00	+2.00000e+00	5.94551e-15
APSO	2.00000e+00	2.00000e+00	+2.00000e+00	1.76109e-08
PSO	2.00000e+00	2.00000e+00	+2.00000e+00	1.96425e-12

To further ascertain the efficiency of *pdAPSO*, more experiments were conducted using the 12 benchmark functions listed in table 4.21. This is because these benchmark functions are popularly accepted as suitable functions in determining the performance of global optimisation algorithms as seen in (Zhan et al. 2011; Huang et al. 2012; Suganthan et

al. 2005; Liang et al. 2005). We have earlier used these functions to measure the performance of *pdPSO* in this chapter.

Table 4.21: Statistical result of 12 benchmarking functions for *pdAPSO*

Algorithm Name	**Primal-Dual-APSO**							
Function Name	Best Fitness	Worst Fitness	Mean Fitness	Standard Deviation	SP	Success Rate (%)	Runtime (s)	NFE
Sphere	-4.50e+002	-4.50e+002	4.50e+002	8.88e-003	332.00	100	9.47	332
Schwefel's P2.22	-4.50e+002	-4.49e+002	-4.50e+002	5.83e-003	495.00	95.76	142.44	532
Rosenbrock	3.90e+002	3.80e+002	3.90e+002	3.01e-002	3321.16	68.58	140.12	2124
Rastrigin	-3.34e+002	5.75e+002	-3.32e+002	4.14e-002	3526.79	99.55	551.27	3545
Ackley	-1.40e+002	1.70e+002	-1.40e+002	5.01e-002	3321.16	88.55	270.89	2124
Schwefel	-4.59e+002	-3.28e+002	-4.59e+002	4.73e-002	8316.73	71.29	647.59	5929
Griewank	-1.80e+002	2.85e+002	-1.83e+002	4.06e-002	1734.56	69.7	259.88	1160
Rotated Rosenbrock	3.89e+002	3.85e+002	3.92e+002	3.09e-002	3380.16	70.15	152.60	2197
Rotated Rastrigin	-3.30e+002	2.18e+002	-3.30e+002	2.94e-002	7301.35	74.19	693.03	5403
Rotated Ackley	-1.40e+002	1.77e+002	-1.42e+002	5.03e-002	3378.16	68.83	296.10	2185
Rotated Griewank	-1.85e+002	2.80e+002	-1.80e+002	4.01e-002	1860.25	70.5	249.33	1105
Shifted Rosenbrock	3.78e+002	3.84e+002	3.85e+002	2.98e-002	3182.72	69.98	148.42	2256
Shifted Rastrigin	-3.30e+002	5.72e+002	-3.31e+002	4.11e-002	3526.38	98.18	543.49	3402

4.5 Performance Comparison of Primal-Dual-PSO (*pdPSO*) and Primal-Dual-APSO (*pdAPSO*)

In this section we present a comparison between the two new algorithms that we proposed in this thesis.

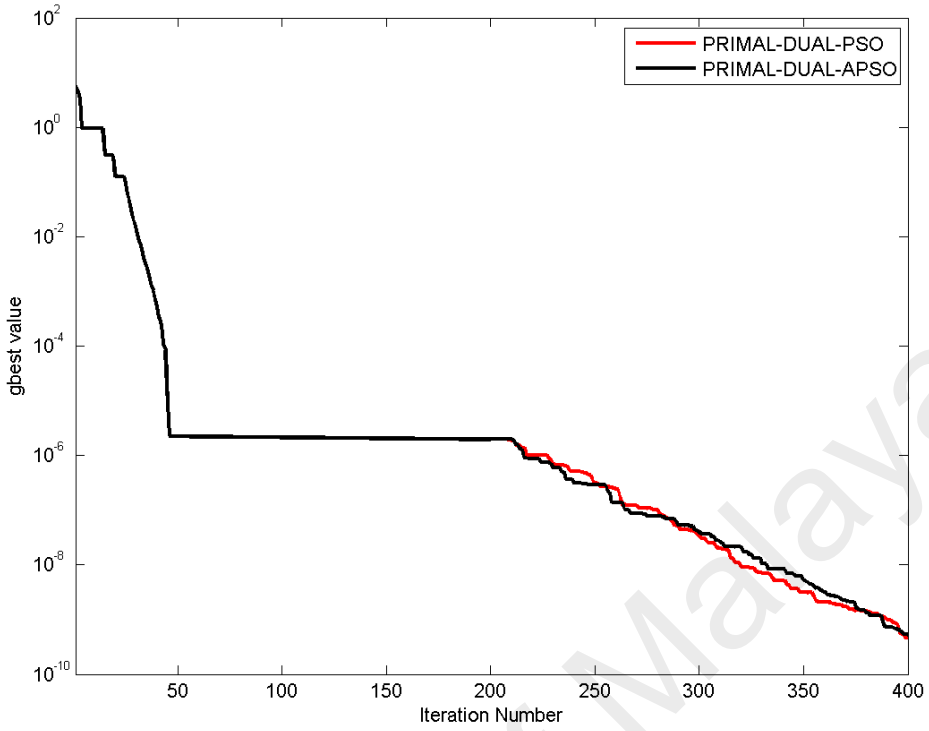


Figure 4.24: Graph of Ackley function for *pdPSO* and *pdAPSO*

Table 4.22: Result Comparison for Ackley function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	5.79464e-10	2.28608e-05	+9.00249e-07	4.16412e-06
Primal-Dual-APSO	5.16059e-10	2.48315e-04	+8.65840e-06	4.52952e-05

The two algorithms converged to global optimum. The convergence rates of *pdPSO* and *pdAPSO* are almost the same. In our comparisons, we used the values of the Best fitness, Mean fitness, and Standard deviation because they are some of the performance measures mentioned in (Chena et al, 2010). The result of our simulation is depicted in Figure 4.25 and Table 4.22 above. When we compared the performance of the two algorithms in terms of the numerical values of Best fitness, Mean fitness, and Standard deviation, we can deduce that the performance of *pdAPSO* is slightly better than that of *pdPSO* algorithm for Ackley function.

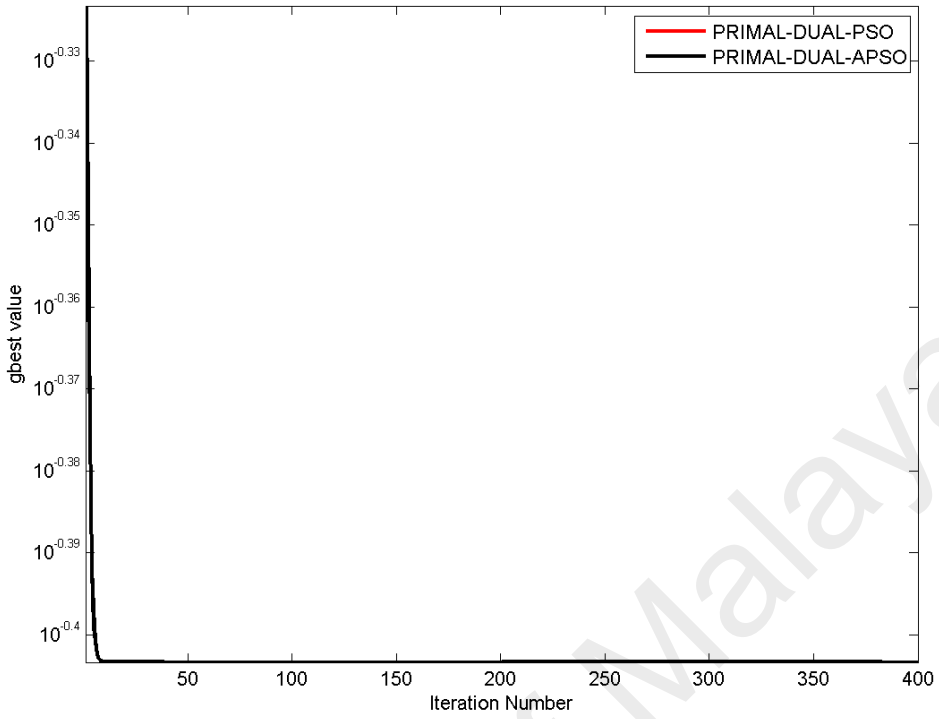


Figure 4.25: Graph of Schaffer f6 modified function for *pdPSO* and *pdAPSO*

Table 4.23: Result Comparison for Schaffer f6 modified function

Algorithm	**Best Fitness**	**Worst Fitness**	**Mean Fitness**	**Std. Deviation**
Primal-Dual-PSO	3.95063e-01	5.66018e-01	+4.03145e-01	3.34147e-02
Primal-Dual-APSO	3.95063e-01	5.70032e-01	+4.08469e-01	4.19347e-02

In term of the numerical value of the Best fitness, the performance of *pdAPSO* and *pdPSO* are almost the same. However, the performance of the *pdPSO* is better in terms of Mean fitness and Standard deviation. The result of our simulation is depicted in Figure 4.26 and Table 4.23 above. The *pdAPSO* and *pdPSO* were able to overcome the problem of premature convergence. Also, the *pdAPSO* and *pdPSO* were able to escape been trapped in the local minima.

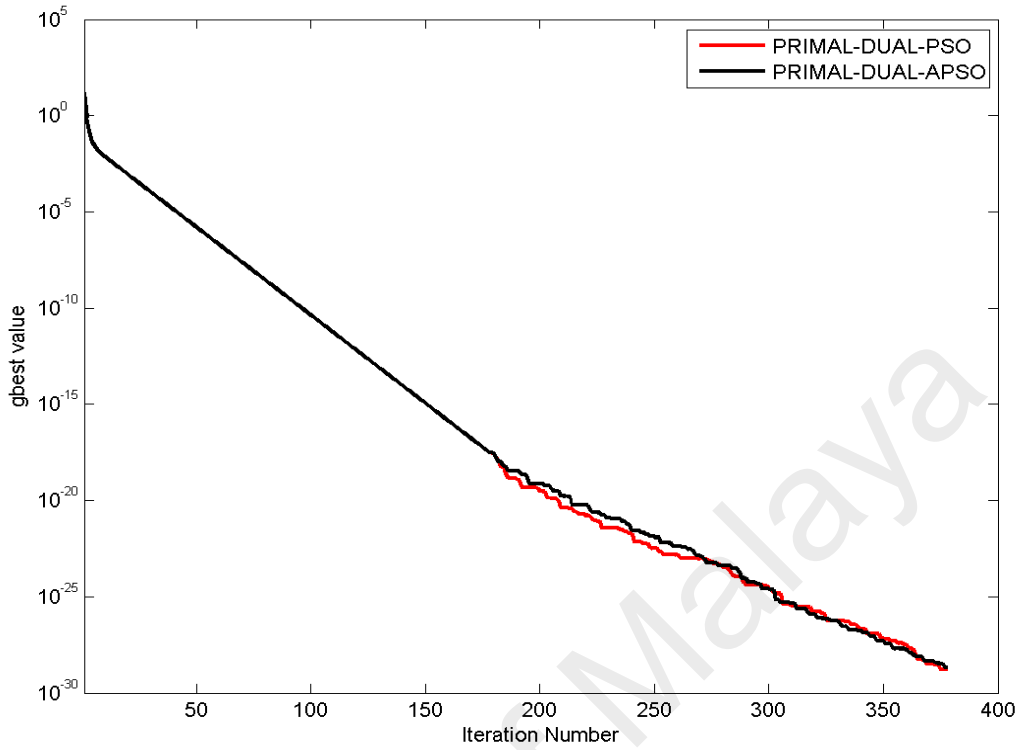


Figure 4.26: Graph of ND Parabola function for *pdPSO* and *pdAPSO*

Table 4.24: Result Comparison for ND Parabola function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	1.87351e-29	9.11685e-27	+1.19780e-27	2.20741e-27
Primal-Dual-APSO	3.30328e-29	2.19468e-25	+9.62640e-27	4.03483e-26

The simulation result above is the NDParabola function for *pdAPSO* and *pdPSO*. The result of our simulation is depicted in Figure 4.27 and Table 4.24 above. The two algorithms converged to global optima, and they all have several local optima with massive drop in the gbest values from the beginning of the iteration to the end as shown in the result. The convergence speed of *pdAPSO* and *pdPSO* are the same. Based on the numerical values of the Best fitness, Mean fitness and Standard deviation, *pdPSO* performs better for this function. Our new algorithm (*pdAPSO*) also demonstrates its ability to escape been trapped

in the local minima and to evade premature convergence in this function. Based on the numerical values of the Best fitness, Mean fitness and Standard deviation, *pdPSO* performs better for this function.

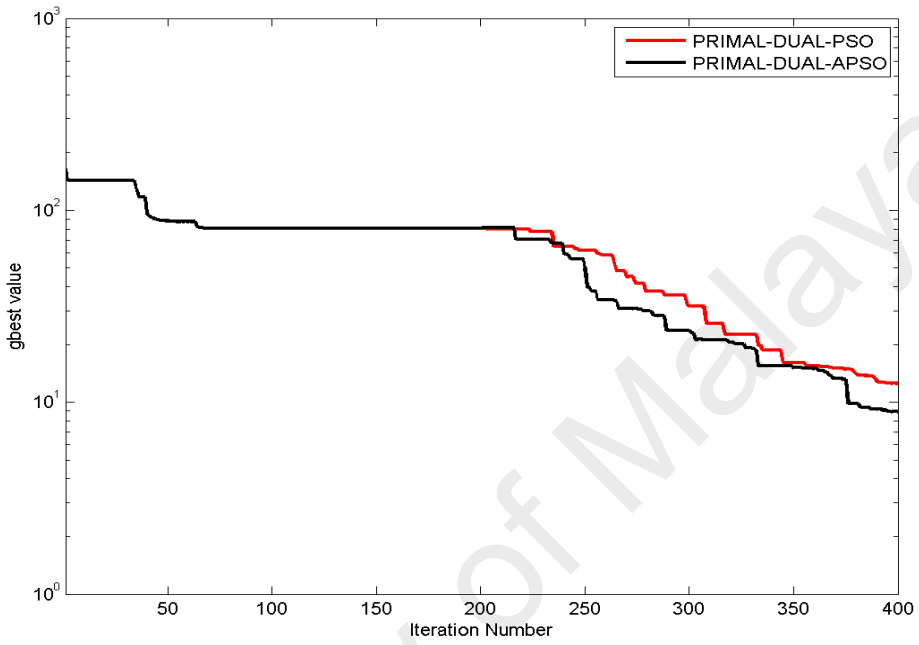


Figure 4.27: Graph of Rastrigin function for *pdPSO* and *pdAPSO*

Table 4.25: Result Comparison for Rastrigin function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	1.25268e+01	2.04591e+02	+4.71626e+01	4.31244e+01
Primal-Dual-APSO	8.87397e+00	7.67439e+01	+2.78492e+01	1.92259e+01

The simulation result above is the Rastrigin function for *pdAPSO* and *pdPSO* as depicted in Figure 2.28 and Table 4.25 above. The two algorithms converged to global optima, and they all have several local optima from the beginning of the iteration to the end as shown in the result. The *pdAPSO* algorithm converged faster than *pdPSO*. Based on the numerical values of the Best fitness, Worst fitness, Mean fitness and Standard deviation, *pdAPSO* performs

better for this function. Our new algorithm (*pdAPSO*) also demonstrates its ability to escape been trapped in the local minima and to escape premature convergence in this function.

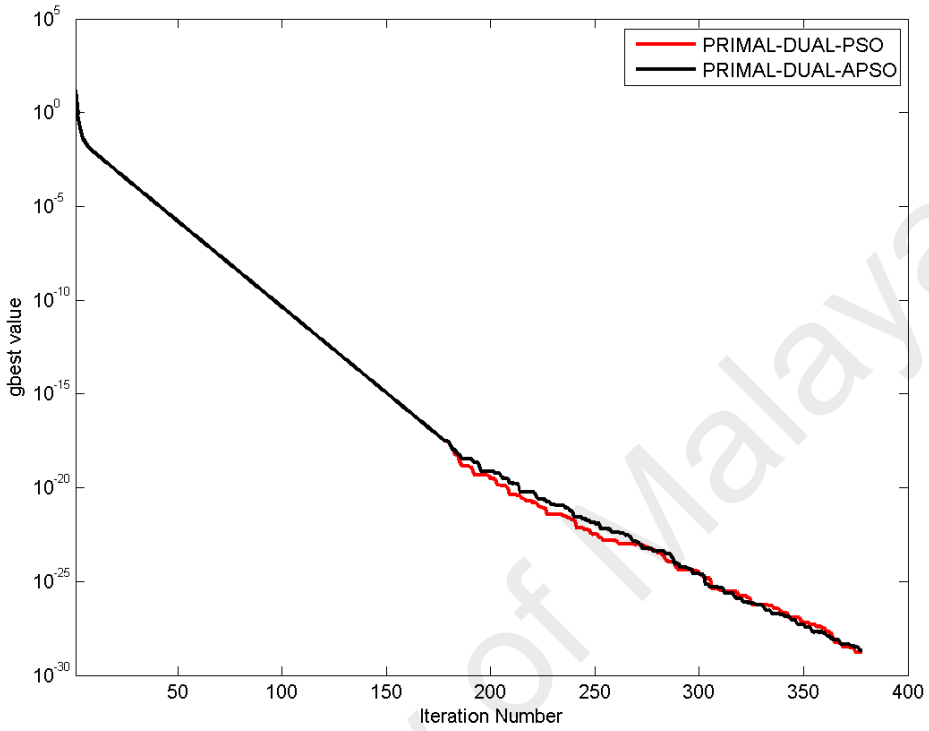


Figure 4.28: Graph of Sphere function for *pdPSO* and *pdAPSO*

Table 4.26: Result Comparison for Sphere function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	1.87351e-29	9.11685e-27	+1.19781e-27	2.20741e-27
Primal-Dual-APSO	3.30324e-29	2.19468e-25	+9.62640e-27	4.03483e-26

From the result of our simulation (depicted as figure 4.29 and table 4.26), there is no significant difference between the performance of *pdAPSO* and *pdPSO* in term of convergence speed. We compared the performance of the two algorithms based on the values of Best fitness, Mean fitness and Standard deviation. From the numerical results, the performance of *pdPSO* was better in terms of the Best fitness, Mean fitness and Standard

deviation.

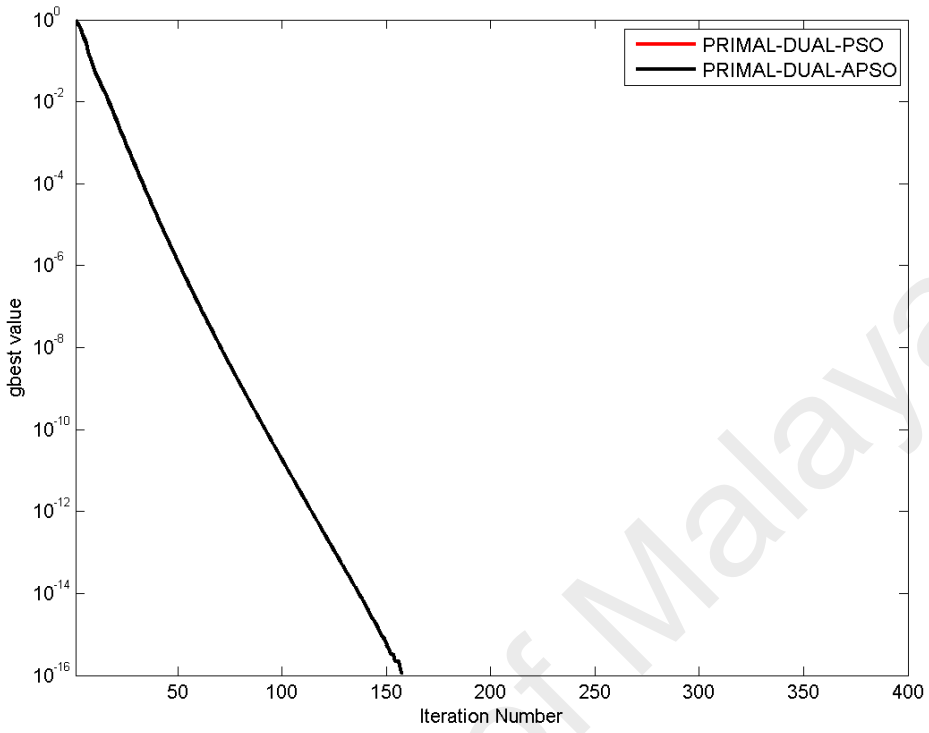


Figure 4.29: Graph of Griewank function for *pdPSO* and *pdAPSO*

Table 4.27: Result Comparison for Griewank function

Algorithm	**Best Fitness**	**Worst Fitness**	**Mean Fitness**	**Std. Deviation**
Primal-Dual-PSO	1.25268e+01	2.04591e+02	+4.71626e+01	4.31244e+01
Primal-Dual-APSO	8.87397e+00	7.67439e+01	+2.78492e+01	1.92259e+01

The result of our simulation is shown in Figure 4.30 and Table 4.27 above. Using the numerical values of the Best fitness and Mean fitness as parameters for our judgment, there was no much difference between the performances of these two algorithms. The performance of *pdPSO* was better based on the numerical value of the Standard deviation when compared to the other three algorithms. From our experiments, *pdAPSO* was able to achieve our aim of designing an algorithm that will overcome the problem of premature convergence that usually characterize the standard PSO and many of its variants.

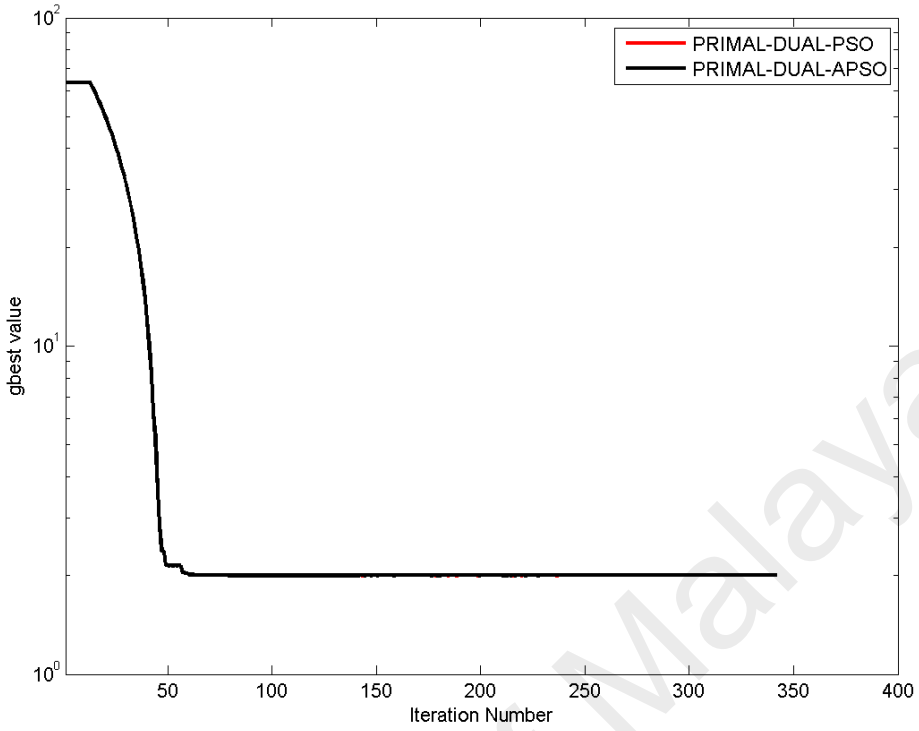


Figure 4.30: Graph of Tripod function for *pdPSO* and *pdAPSO*

Table 4.28: Result Comparison for Tripod function

Algorithm	Best Fitness	Worst Fitness	Mean Fitness	Std. Deviation
Primal-Dual-PSO	2.00000e+00	2.00000e+00	+2.00000e+00	5.94551e-15
Primal-Dual-APSO	2.00000e+00	2.00000e+00	+2.00000e+00	9.72321e-15

Simulation result of Tripod function for *pdPSO* and *pdAPSO* is presented in Figure 4.30 and Table 4.28 above. Using the numerical values of the Best fitness and Mean fitness as parameters for our judgment, there was no much difference between the performances of these two algorithms. The performance of *pdPSO* was better based on the numerical value of the Standard deviation when compared to the other three algorithms. From our experiments, *pdAPSO* was able to achieve our aim of designing an algorithm that will overcome the

problem of premature convergence that usually characterize the standard PSO and many of its variants.

With reference to convergence speed, *pdAPSO* was faster than *pdPSO* in 5 functions out of the total of 7 functions that we considered. We can therefore consider *pdAPSO* as a fast algorithm that can be used to solve complex numerical optimisation problems. The *pdAPSO* also have a higher level of steadiness in comparison to the other three algorithms. The values of Mean fitness and Standard deviations for Sphere, NDParabola and Tripod functions were very small when compared to the ones of *pdPSO*, PSO and APSO. We can therefore conclude that *pdAPSO* is a very stable algorithm that has the capacity to produce rational results that are reliable. Finally we can deduce that *pdAPSO* is a robust algorithm as it performs better than *pdPSO*, PSO and APSO in its ability to successfully find the global optimum on all the benchmarking functions we used especially on Griewank, Schaffer f6 modified, NDParabola and Rastrigin functions which prove to be very problematic to solve for many modern day optimisation algorithms. Consequently, *pdAPSO* can be considered as a robust algorithm that can withstand difficulties during optimisation process.

4.6 Performance Comparison of *pdPSO* and *pdAPSO* algorithms with the state-of-the-art PSO variants

In this section we compared the performance of *pdPSO* and *pdAPSO* with nine (9) state of the art algorithms as listed in the table below. The conventional PSO algorithm that has been popularly applied in different field is PSO-LDIW which was proposed by Shi and Eberhart (1999). The comprehensive learning strategy PSO (CLPSO) was proposed by Liang et al. (2006) with the purpose of producing superior performance compared to the existing PSO variants for multimodal functions. The Perturbed particle swarm optimisation

for numerical optimisation was (pPSA) was proposed by Zhao (2010). The algorithms device a strategy for handling premature convergence by employing a particle updating approach that centres on the idea of perturbed global best particle. The rank based particle swarm optimisation algorithm with dynamic adaptation (PSO_{rank}) was proposed by Akbari and Ziarati (2011). The algorithm exploits the collaborative behavior of particles to make a meaningful increase in the efficiency of the conventional PSO algorithm. Zhan et al. (2011) proposed the orthogonal learning PSO (OLPSO-G). This algorithm uses a perpendicular learning approach to create a favourable and effective model to pilot particles to move in most suitable directions. Huang et al. (2012) developed the Example-based learning PSO (ELPSO) for continuous optimisation. Their purpose is to use example-based learning scheme to proffer a superior performance for multimodal functions. An adaptive parameter tuning of PSO centered on velocity information (APSO-VI) algorithm was proposed by Xu (2013). Diversity enhanced PSO with neighbourhood (DNSPSO) was presented by Wang et al. (2013). This algorithm engages the variety improving method and neighborhood search tactics to attain a swapping between exploration and exploitation. Multiobjective sorting-based learning PSO for continuous optimisation (MLPSO) proposed by Gang et al. (2016) uses the MSL approach to direct particles to move in the most suitable path by creating a direction paradigm that have superior fitness value and variety in swarm population. The parameter settings for these PSO variations are specified in Table 4.29 with reference to their references. The purpose of using these PSO variants for our comparisons is because they are state of the art PSO algorithms which cover a broad period of time from 1999 to 2016. Furthermore, they have been described in literature as high performing variants of PSO with reference to their experimented problems.

Table 4.29: PSO variants used for our comparative studies.

PSO variants	Parameter Setting	Reference
PSO-LDIW	w : 0.9–0.4, c1 = c2 =2	Shi and Eberhart (1999)
CLPSO	w : 0.9–0.4, c = 1.49, m =7	Liang et al. (2006)
pPSA	w = 0.9, $c_1 = 0.5$, $c_2 = 0.3$, $\sigma_{max} = 0.15$, $\sigma_{min} = 0.001$, $\propto = 0.5$	Zhao (2010)
PSO_{rank}	w is non-linear, $\propto = 0.45$, $\beta = 0.385$, m = 2	Akbari and Ziarati (2011)
OLPSO-G	w: 0.9–0.4, c = 2.0, G = 5, V_{max} = 0.2 9 x Range	Zhan et al. (2011)
ELPSO	w = 0.729, c = 1.49, m = 4	Huang et al. (2012)
APSO-VI	w:0.9–0.3, $c_1 = c_2 = 1.49$	Xu (2013)
DNSPSO	w = 0.729, c1 = c2 = 1.49618, k = 2, p_r = 0.9, p_{ns} = 0.6	Wang et al. (2013)
MLSPSO	w:0.9–0.4, $c_1 = c_2 = 2$	Gang, et al (2016)

From the experiments that were conducted, the algorithm configurations of *pdPSO* and *pdAPSO* are as follows. The inertia weight w is linearly decreasing from 0.9 to 0.4, and c_1 and c_2 are set to 1.49. For a fair comparison among all the PSO variants, the population size is set at 50 and the maximum fitness evaluations (FEs) is set at 30,000. We carried out experiment 30 times for each algorithm using twelve (12) benchmarking functions and the statistical values of the Best Fitness, Worst Fitness, Mean Fitness, Standard Deviation, SP, Success Rate (%), Runtime (s), and NFE are used in the evaluations.

4.6.1 Performance Comparison on superiority of results

We make comparison of the performance of the PSO algorithms listed in table 4.29 with that of *pdPSO* and *pdAPSO*. The results of our comparison are in tables 4.30 – 4.42 where we compared the mean and standard deviations of the eleven (11) algorithms. The best results obtained among the other eleven algorithms we evaluated their performance are boldfaced. The first three functions (Sphere, Schwefel's P2.22, and Rosenbrock) we considered are unimodal functions. The first two are comparatively easy and virtually all the algorithms can solve them. The two algorithms that proffer the best results for Sphere are

pdPSO and ELPSO while *pdPSO*, *pdAPSO* and OLPSO-G proffers the best results for Schwefel's P2.22. For Rosenbrock function, *pdAPSO* and MSLPSO proffers the best solution. This function is used to test the ability of an algorithm to solve a hard problem because it contains very narrow valley in its landscape. It is only these two algorithms that were able to escape being trapped in its local optima.

Table 4.30: Mean and Standard Deviation comparisons for sphere among eleven (11) PSO algorithms

Algorithm	**Mean**	**Standard Deviation**
PSO-LDIW	4.68E-23	8.33E-23
CLPSO	5.23E-14	3.66E-14
pPSA	2.76E-07	5.93E-07
PSOrank	3.91E-09	7.87E-09
OLPSO-G	6.21E-52	2.19E-52
ELPSO	3.38E-94	**1.22E-94**
APSO-VI	1.37E-12	8.39E-12
DNSPSO	8.27E-85	3.69E-85
MSLPSO	2.73E-82	1.69E-82
pdPSO	**-4.50E+00**	6.38E-01
pdAPSO	4.50E+00	8.88E-00

Table 4.31: Mean and Standard Deviation comparisons for Schwefel's P2.22 among eleven (11) PSO algorithms

Algorithm	**Mean**	**Standard Deviation**
PSO-LDIW	4.08E-09	1.09E-09
CLPSO	2.81E-07	3.57E-07
pPSA	2.35E-09	4.94E-09
PSOrank	3.73E-12	5.22E-11
OLPSO-G	3.77E228	**6.77E-28**
ELPSO	8.08E-24	2.99E-24
APSO-VI	4.66E-14	1.44E-14
DNSPSO	7.97E-26	5.99E-26
MSLPSO	1.35E-16	2.98E-16
pdPSO	**-4.50E+00**	3.68E-00
pdAPSO	**-4.50E+00**	5.83E-00

Table 4.32: Mean and Standard Deviation comparisons for Rosenbrock among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	5.59E+01	3.83E+01
CLPSO	2.25E+01	1.21E+01
pPSA	3.57E+01	2.56E+01
PSOrank	4.44E+01	3.14E+01
OLPSO-G	2.51E+01	1.77E+01
ELPSO	1.78E+01	1.59E+01
APSO-VI	1.50E+01	1.23E+01
DNSPSO	7.38E+00	8.82E+00
MSLPSO	2.90E-01	**3.72E-01**
pdPSO	3.92E+00	5.08E+00
pdAPSO	**-3.32E+00**	4.14E-00

The second category of experiments we carried out was on multimodal functions. The Primal Dual method provides PSO the capacity to explore the search space better and exploit the particle in the swarm to its advantage thereby producing enhanced fitness value and create diversity in the swarm population. It is anticipated that *pdPSO* and *pdAPSO* will escape from being trapped in local minima and produce superior results on multimodal functions. For the functions Rastrigin, Ackley, Schwefel, and Griewank, *pdPSO* and *pdAPSO* converged to the global optimum. The close to global optima was attained by *pdPSO* and *pdAPSO* on Rastrigin function. The best result was however produced by *pdPSO* and MSLPSO. The algorithms *pdAPSO* and MSLPSO produced the best result on Ackley function. For the Schwefel function, *pdAPSO* and ELPSO achieved the best solution. On the Griewank function, the best solution is obtained for *pdAPSO* and MSLPSO algorithms. The results of our tests demonstrated that *pdPSO* and *pdAPSO* possess that ability to effectively handle premature convergence problem and escape from being trapped in local minima on majority of the multimodal functions. The successful attainment of global optima solutions on many of the multimodal functions indicates that the performances of *pdPSO* and *pdAPSO* algorithms have really been enhanced through the fusion of Primal-Dual method and PSO algorithm.

Table 4.33: Mean and Standard Deviation comparisons for Rastrigin among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	1.85E+01	2.72E+01
CLPSO	3.98E-08	4.54E-08
pPSA	3.07E-03	6.25E-02
PSOrank	1.08E-12	9.74E-11
OLPSO-G	8.25E-02	4.81E-02
ELPSO	2.89E-14	3.18E-14
APSO-VI	3.82E+00	4.69E+00
DNSPSO	7.66E-15	3.38E-15
MSLPSO	2.37E-15	**1.44E-15**
pdPSO	**-3.30E+00**	4.19E-01
pdAPSO	-1.40E+00	5.01E-00

Table 4.34: Mean and Standard Deviation comparisons for Ackley among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	1.04E-04	3.85E-03
CLPSO	3.00E-11	2.97E-12
pPSA	8.92E-07	7.92E-07
PSOrank	7.82E-10	5.91E-10
OLPSO-G	1.33E-14	5.11E-14
ELPSO	7.69E-15	8.44E-14
APSO-VI	6.77E-14	7.35E-14
DNSPSO	1.89E-14	2.17E-14
MSLPSO	7.23E-16	**2.94E-16**
pdPSO	3.85E+00	5.05E+00
pdAPSO	**-1.40E+00**	5.01E-00

Table 4.35: Mean and Standard Deviation comparisons for Schwefel among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	2.98E+03	9.29E+02
CLPSO	3.86E-03	4.19E-03
pPSA	2.58E+03	6.23E+02
PSOrank	2.32E+03	5.12E+02
OLPSO-G	6.34E+02	8.09E+01
ELPSO	6.56E-03	**3.24E-03**
APSO-VI	2.43E+01	9.49E+00
DNSPSO	5.95E+00	7.17E+00
MSLPSO	9.36E+00	5.44E+00
pdPSO	-4.43E+00	4.07E+00
pdAPSO	**-4.59E+00**	4.73E-00

Table 4.36: Mean and Standard Deviation comparisons for Griewank among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	1.84E-04	2.77E-04
CLPSO	2.88E-09	6.47E-08
pPSA	9.02E-06	7.29E-06
PSOrank	1.54E-04	4.24E-03
OLPSO-G	3.41E-03	1.03E-03
ELPSO	9.78E-23	3.11E-23
APSO-VI	4.88E-12	2.07E-11
DNSPSO	3.96E-38	2.31E-38
MSLPSO	5.91E-43	**1.38E-43**
pdPSO	-1.80E+00	1.08E-00
pdAPSO	**-1.83E+00**	4.06E-00

Finally, we investigated the performances of the eleven algorithms on rotated and shifted functions. Rotated Rosenbrock, Rotated Rastrigin, Rotated Ackley, and Rotated Griewank are multimodal functions with rotated coordinates. The results of our algorithm test on the rotated functions and that of shifted functions are presented in Tables 4.37 – 4.42. The *pdPSO* and *pdAPSO* algorithms attained global optima for all the rotated functions. On Rotated Rosenbrock function, *pdAPSO* produced the best result. MSLPSO and *pdAPSO* achieved the best result for Rotated Rastrigin function. ELPSO and *pdAPSO* have the best result for the Rotated Ackley function. For Rotated Griewank function, *pdPSO* and MSLPSO achieved the best solution. It should be noted that the rotation does not affect the performance of the *pdAPSO* and *pdPSO*. Infact the effectiveness of our algorithms become more pronounced with test on all the rotated functions especially Rotated Rosenbrock where *pdAPSO* produced the most accurate result and closely followed by *pdPSO* and MSLPSO respectively. To be precise, our experiments confirmed the observation of Wang et al. (2012) that the Rotated Rosenbrock function proved very difficult for other PSO algorithms to escape being trapped in its local optima as the function becomes more problematic after rotating its coordinates.

Table 4.37: Mean and Standard Deviation comparisons for Rotated Rosenbrock among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	8.89E+01	7.27E+01
CLPSO	5.27E+01	3.88E+01
pPSA	6.06E+01	5.25E+01
PSOrank	5.87E+01	6.29E+01
OLPSO-G	3.46E+01	3.82E+01
ELPSO	3.13E+01	2.44E+01
APSO-VI	4.51E+01	1.78E+01
DNSPSO	2.92E+01	2.13E+01
MSLPSO	6.38E+00	5.45E+00
pdPSO	3.99E+00	5.54E+00
pdAPSO	**3.92E+00**	**3.09E-00**

Table 4.38: Mean and Standard Deviation comparisons for Rotated Rastrigin among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	9.26E+01	8.73E+01
CLPSO	4.23E+01	3.78E+01
pPSA	6.78E+01	5.33E+01
PSOrank	5.24E+01	4.19E+01
OLPSO-G	2.81E+01	2.21E+01
ELPSO	1.62E+00	3.92E-01
APSO-VI	1.23E+01	8.24E+00
DNSPSO	6.45E-14	7.19E-14
MSLPSO	5.89E-15	**8.82E-15**
pdPSO	-1.80E+00	1.08E-00
pdAPSO	**-3.30E+00**	2.94E-00

Table 4.39: Mean and Standard Deviation comparisons for Rotated Ackley among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	9.55E+00	7.67E+00
CLPSO	7.21E-05	6.24E-04
pPSA	6.34E-04	9.35E-04
PSOrank	3.27E-06	8.47E-05
OLPSO-G	2.93E-13	1.79E-12
ELPSO	9.73E-14	**2.04E-14**
APSO-VI	4.85E-05	5.25E-05
DNSPSO	5.89E-14	4.75E-14
MSLPSO	8.68E-14	7.34E-14
pdPSO	-1.39E+00	5.51E-00
pdAPSO	**-1.42E+00**	5.03E-00

Table 4.40: Mean and Standard Deviation comparisons for Rotated Griewank among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	9.68E-02	6.33E-01
CLPSO	4.20E-08	2.72E-09
pPSA	5.14E-04	3.23E-04
PSOrank	1.52E-03	2.99E-03
OLPSO-G	4.08E-03	3.88E-03
ELPSO	5.06E-13	4.24E-13
APSO-VI	1.35E-06	1.03E-05
DNSPSO	3.98E-21	4.34E-22
MSLPSO	2.17E-35	**8.92E-34**
pdPSO	**-1.82E+00**	1.06E-00
pdAPSO	-1.80E+00	4.01E-00

The outcome of our experiments also indicates that *pdPSO* and *pdAPSO* compete very well with other state of the art algorithms. On the Shifted Rosenbrock functions, *pdPSO* produced the best result and closely followed by *pdAPSO* and MSLPSO respectively. The other PSO algorithms are trapped in local optima this function. For Shifted Rastrigin function, *pdAPSO* produced the most accurate result and closely followed by *pdPSO* and OLPSO-G respectively. In summary, the rotation and shift affected the performance of the other nine algorithms while the efficiency of *pdPSO* and *pdAPSO* becomes more noticeable with the rotation and the shift. The comparisons reveal that the integration of Primal-Dual into PSO is advantageous to enhancing the performance of PSO. We hereby conclude that *pdPSO* and *pdAPSO* have a superior performance compared to the other PSO variants on majority of the rotated functions and on the two shifted functions.

Table 4.41: Mean and Standard Deviation comparisons for Shifted Rosenbrock among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	6.15E+02	9.58E+01
CLPSO	4.87E+02	3.36E+01
pPSA	5.29E+02	8.53E+01
PSOrank	5.62E+02	7.87E+01
OLPSO-G	4.63E+02	4.63E+01
ELPSO	4.26E+02	4.04E+01
APSO-VI	4.41E+02	3.52E+01
DNSPSO	4.48E+02	3.27E+01
MSLPSO	4.13E+02	3.11E+01
pdPSO	**-3.30E+00**	**4.19E-01**
pdAPSO	3.85E+00	2.98E-00

Table 4.42: Mean and Standard Deviation comparisons for Shifted Rastrigin among eleven (11) PSO algorithms

Algorithm	Mean	Standard Deviation
PSO-LDIW	-1.38E+02	3.65E+01
CLPSO	-3.07E+02	7.44E+00
pPSA	-2.78E+02	1.39E+00
PSOrank	-2.42E+02	1.84E+00
OLPSO-G	-3.26E+02	2.33E+00
ELPSO	-3.03E+02	5.66E+00
APSO-VI	-2.89E+02	9.24E+00
DNSPSO	-3.11E+02	7.17E+00
MSLPSO	-3.22E+02	6.34E+00
pdPSO	-2.97E+00	7.18E+00
pdAPSO	**-3.31E+00**	**4.11E-00**

4.6.2 Performance Comparison on the dependability and speed of convergence

The dependability of an algorithm is a determined by the mean of success rate on the entire test functions. The convergence speed in attaining the global optimum is also a striking standard for determining the performance of any optimisation algorithm. The rates of success of all eleven variants of PSO algorithm on individual test function and the dependability of the algorithms are shown in Tables 4.42 – 4.54 below.

Table 4.43: Comparison of dependability and speed of convergence on Sphere

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	5571	**100**	5571
CLPSO	7069	**100**	7069
pPSA	6954	**100**	6954
PSOrank	6631	**100**	6631
OLPSO-G	3872	**100**	3872
ELPSO	4396	**100**	4396
APSO-VI	24,910	**100**	24,910
DNSPSO	4767	**100**	4767
MSLPSO	5853	**100**	5853
pdPSO	1078	**100**	1078
pdAPSO	**332**	**100**	**332**

Table 4.44: Comparison of dependability and speed of convergence on Schwefel's P2.22

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	12,049	**100**	12,049
CLPSO	14,702	**100**	14,702
pPSA	11,205	**100**	11,205
PSOrank	9929	**100**	9929
OLPSO-G	9301	**100**	9301
ELPSO	8973	**100**	8503
APSO-VI	18,932	**100**	18,932
DNSPSO	10,345	**100**	10,345
MSLPSO	8828	**100**	8828
pdPSO	2996	91.7	3268.36
pdAPSO	**495.00**	95.7	**532**

Table 4.45: Comparison of dependability and speed of convergence on Rosenbrock

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	8938	**100**	8938
CLPSO	6786	**100**	6786
pPSA	10,080	**100**	10,080
PSOrank	9607	**100**	9607
OLPSO-G	11,680	**100**	11,680
ELPSO	9287	**100**	9287
APSO-VI	23,940	**100**	23,940
DNSPSO	6269	**100**	6269
MSLPSO	6198	**100**	6198
pdPSO	2997	80.6	8934.35
pdAPSO	**2124**	68.58	**3321.16**

Table 4.46: Comparison of dependability and speed of convergence on Rastrigin

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	12,164	**100**	12,164
CLPSO	15,423	**100**	15,423
pPSA	11,835	**100**	11,835
PSOrank	10,387	**100**	10,387
OLPSO-G	9629	**100**	9629
ELPSO	9302	**100**	9302
APSO-VI	27,127	**100**	27,127
DNSPSO	10,073	**100**	10,073
MSLPSO	7896	**100**	7896
pdPSO	**2827**	**100**	**2827**
pdAPSO	3545	99.55	3526.79

Table 4.47: Comparison of dependability and speed of convergence on Ackley

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	11,216	16.67	67,323
CLPSO	12,957	**100**	12,957
pPSA	13,720	**100**	13,720
PSOrank	13,142	**100**	13,142
OLPSO-G	10,751	**100**	10,751
ELPSO	6698	**100**	6698
APSO-VI	29,361	**100**	29,361
DNSPSO	10,673	**100**	10,673
MSLPSO	12,992	**100**	12,992
pdPSO	8285	85.7	8821.14
pdAPSO	**2124**	88.55	**3321.16**

Table 4.48: Comparison of dependability and speed of convergence on Schwefel

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	18,243	30.33	60,148
CLPSO	10,024	**100**	10,024
pPSA	15,045	43.33	34,722
PSOrank	13,956	76.67	18,205
OLPSO-G	9827	**100**	9827
ELPSO	9306	**100**	9306
APSO-VI	23,194	**100**	23,194
DNSPSO	10,083	**100**	10,083
MSLPSO	9362	**100**	9362
pdPSO	**5615.45**	58.63	**2981**
pdAPSO	5929	71.29	8316.73

Table 4.49: Comparison of dependability and speed of convergence on Griewank

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	7781	40	19,452
CLPSO	9204	**100**	9204
pPSA	13,567	**100**	13,567
PSOrank	8318	**100**	8318
OLPSO-G	7432	63.33	11,735
ELPSO	7672	**100**	7672
APSO-VI	28,381	**100**	28,381
DNSPSO	8315	**100**	8315
MSLPSO	7159	**100**	7159
pdPSO	2573	41.66	6175.20
pdAPSO	**1160**	69.7	**1734.56**

Table 4.50: Comparison of dependability and speed of convergence on Rotated Rosenbrock

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	9897	33.33	29,694
CLPSO	10,623	76.67	13,857
pPSA	14,929	63.33	23,573
PSOrank	13,057	76.67	17,032
OLPSO-G	12,958	93.33	13,884
ELPSO	10,034	**100**	10,034
APSO-VI	27,035	73.33	36,868
DNSPSO	9836	**100**	9836
MSLPSO	8742	**100**	8742
pdPSO	3456	38.79	8986.74
pdAPSO	**2197**	70.15	**3380.16**

Table 4.51: Comparison of dependability and speed of convergence on Rotated Rastrigin

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	14,278	76.67	18,625
CLPSO	16,085	**100**	16,085
pPSA	12,537	**100**	12,537
PSOrank	11,043	**100**	11,043
OLPSO-G	10,074	**100**	10,074
ELPSO	9737	**100**	9737
APSO-VI	11,003	**100**	11,003
DNSPSO	9265	**100**	9265
MSLPSO	8792	**100**	8792
pdPSO	**2573**	41.66	**6175.20**
pdAPSO	5403	74.19	7301.35

Table 4.52: Comparison of dependability and speed of convergence on Rotated Ackley

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	–	0	–
CLPSO	24,727	40	61,817
pPSA	16,296	16.67	97,815
PSOrank	15,292	80	19,115
OLPSO-G	12,707	**100**	12,707
ELPSO	9318	**100**	9318
APSO-VI	27,824	36.67	75,897
DNSPSO	13,239	**100**	13,239
MSLPSO	9153	**100**	9153
pdPSO	2455	72.19	3748.29
pdAPSO	**2185**	68.83	**3378.16**

Table 4.53: Comparison of dependability and speed of convergence on Rotated Griewank

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	8241	13.33	61,823
CLPSO	12,514	**100**	12,514
pPSA	9574	60	15,957
PSOrank	8933	43.33	20,616
OLPSO-G	9404	33.33	28,215
ELPSO	9242	**100**	9242
APSO-VI	297,10	**100**	29,710
DNSPSO	5981	**100**	5981
MSLPSO	7748	**100**	7748
pdPSO	1124	68.8	**1734.92**
pdAPSO	**1105**	70.5	1860.25

Table 4.54: Comparison of dependability and speed of convergence on Shifted Rosenbrock

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	9235	13.33	69,280
CLPSO	13,024	43.33	30,058
pPSA	9783	23.33	47,351
PSOrank	12,312	16.67	73,902
OLPSO-G	10,331	66.67	15,496
ELPSO	10,376	73.33	14,149
APSO-VI	26,292	83.33	31,552
DNSPSO	8832	86.67	10,192
MSLPSO	9783	**100**	9783
pdPSO	2827	**100**	**2827**
pdAPSO	**2256**	69.98	3182.72

Table 4.55: Comparison of dependability and speed of convergence on Shifted Rastrigin

Algorithm	Mean FEs	SR%	SP
PSO-LDIW	11,097	26.66	41,624
CLPSO	13,295	**100**	13,295
pPSA	11,092	53.33	20,799
PSOrank	10,551	63.33	16,660
OLPSO-G	9012	**100**	9012
ELPSO	10,075	**100**	10,075
APSO-VI	28,039	76.67	36,575
DNSPSO	9923	**100**	9923
MSLPSO	12,081	**100**	12,081
pdPSO	**2999**	59.83	5873.29
pdAPSO	3402	98.18	**3526.38**

From the results of our experiment depicted in Tables 4.43 – 4.55, the mean dependability of *pdAPSO* is 80.4% while that of *pdPSO* is 69.69%. This is an indication that the fusion of Primal Dual method and PSO will increase the dependability to PSO in overcoming premature convergence and converging to global optima. The MSLPSO have the highest percentage of dependability across all the benchmark functions used in our experiment with the mean dependence of 97.71%. This is because the MSL approach offers a better direction for the particles to move to a promising area in the search space (Gang, 2016). It is worthy of note that *pdPSO* and *pdAPSO* converged at the global optimum for all the test functions. PSO-LDIW was unable to converge on Rotated Ackley function. The ratio of dependability of *pdPSO* and *pdAPSO* indicated that our algorithms offer a dependable and robust method for providing solution to global optimisation problems.

The pace at which an algorithm attains the global optimum is a very important parameter for assessing the performance of the algorithm. Since the Primal Dual method is a robust optimisation algorithm, it is expected that *pdPSO* and *pdAPSO* will produce superior result in comparison to so other state of the art algorithm with a better speed of convergence.

To substantiate our claim, the results of Mean FEs and SP, for the eleven algorithms are shown in Tables 4.43 – 4.55. The best results are boldface in each of the Tables.

It is very obvious from those tables that the speed of convergence of *pdAPSO* and *pdPSO* algorithms is superior to the other PSO algorithms on all the benchmark functions. For instance, on Schwefel's function, the mean FEs of 5571, 7069, 6954, 6631, 3872, 4396, 24,910, 4767, and 5853 are required by PSO-LDIW, CLPSO, pPSA, PSOrank, OLPSO-G, ELPSO, APSO-VI, DNSPSO and MSLPSO respectively to attain the global optima. However, *pdPSO* and *pdAPSO* only use 1078 and 332 respectively which is an indication that *pdAPSO* is the fastest while *pdPSO* is second to it. To be concise, *pdPSO* and *pdAPSO* uses the lowest number of FEs to attain satisfactory solutions for all the 12 benchmark functions. This is another confirmation that the Primal Dual method has enhance the PSO algorithm in producing better fitness value and creating diversity in the swarm population to improve the convergence speed of PSO particles.

4.7 Chapter summary

This chapter presents two new hybrid algorithm optimisation algorithm named Primal Dual Interior Point Method Particle Swarm Optimisation (*pdPSO*) and Primal Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*). These algorithm combines the explorative ability of PSO with the exploitative capacity of the Primal Dual Interior Point Method thereby possessing a strong capacity of avoiding premature convergence since it combines the strength of both the Primal-dual method and the PSO algorithm. The hybrid method increases the effectiveness of the PSO method and the Primal-Dual method by speedy convergence and improved value of objective function. A comparative study of the proposed algorithm has been conducted with the conventional PSO and Primal Dual method using nine benchmark functions. It is very clear that our algorithm performs better in terms of precision,

rate of convergence, steadiness and robustness. The behaviour of *pdPSO* under the unimodal and multimodal functions shows that the algorithm will be a suitable tool in solving complicated optimisation problems that PSO alone or Primal Dual alone cannot solve efficiently. In the next chapter, we are going to present another novel algorithm called Primal Dual Asynchronous Particle Swarm Optimisation.

Several experiments were conducted; firstly, a comparison of *pdPSO* algorithm with the typical PSO, APSO and *pdPSO* using nine (9) benchmark functions was done. Secondly, we compared the performance of *pdAPSO* with the conventional PSO, APSO and *pdPSO* using seven (7) benchmark functions was done. The performances of our algorithms (*pdPSO* and *pdAPSO*) are better in terms of precision, rate of convergence, steadiness and robustness when compared to some variants of PSO. The behaviour of *pdPSO* and *pdAPSO* under the unimodal and multimodal functions shows that the algorithms will be suitable tool in solving complicated optimisation problems that PSO alone or Primal Dual alone cannot solve efficiently. The result of our experiments demonstrates that there is little improvement in the performance of *pdAPSO* compared to that of *pdPSO* based on the statistical results of the best fitness, mean fitness and standard deviation.

Thirdly, we compared the performance of *pdPSO* and *pdAPSO* with nine other state of the art algorithms using 12 benchmark functions. We did a comparison of the performance and superiority of solutions of the 11 algorithms, and the outcome of our tests show that *pdPSO* and *pdAPSO* have the capacity to overcome the problem premature convergence and prevent particles from being trapped in local minima on many all the functions. The comparison of dependability and speed of convergence of the 11 algorithms on 12 benchmark functions was also done. The result of our experiment shows that *pdPSO* and *pdAPSO* are reliable and robust algorithms for solving global optimisation problems. The convergence speed of *pdAPSO* and *pdPSO* algorithms were compared to the other state of the art PSO

algorithms and our proposed algorithms proved to attain the global optima on all the benchmark functions in the shortest run time.

CHAPTER 5: APPLICATIONS OF PRIMAL-DUAL-PSO TO SWARM ROBOTICS TASKS

5.1 Introduction

In this chapter, we applied one of our proposed algorithms (*pdPSO)* to solve the problem of flocking and pattern formation in swarm robotics. The problem synonym with the field of swarm robotics is managing and directing the movement of a considerable number of robots to carry out a mission together. This type of task is normally impracticable, demanding and laborious for a particular set of robots to accomplish. The inspiration of swarm robots is fundamentally drawn from the study of behaviour of animals like the flock of birds, herd of cattle, and shoal of fish. According to (Bayindir & Şahin, 2007) and (Şahin, 2005), the performance of the swarm at the global level will be largely influenced by the performance of the individual agent at the local level.

Some of the traits of swarm robotics that have been extensively investigated are convergence, foraging (Bayindir & Şahin, 2007), pattern formation (Vicsek et al., 1995), flocking, aggregation and segregation (Reynolds, 1987), box-pushing (Şahin, 2005), cooperative mapping (Jadbadaie, Lin, & Morse, 2003), soccer tournaments (Yang, Xiong, Chong, Défago, 2008), site preparation (Kim, Wang, Shin, 2006), and sorting (Jeschke, Liu & Schilberg, 2011). From the list of the various attributes of swarms above, flocking is the most attractive; where potential practical applications in areas like search and rescue, system for monitoring behaviour or changing information, system for acquiring data which is used for measuring physical phenomenon, and networks of small low cost sensors (Jeschke, Liu & Schilberg, 2011) can be realized.

There are many advantages to be derived from controlling the movement of a group of robots rather than moving the robots one after the other. When robots move in swarm, their sensing capacity is increased. This can result into better performance for carrying out critical missions and collecting data at a specific location in an environment.

5.2 Flocking

When a group of aerial or submarine robots flock in unison, they can generate an energy timesaving movement, just like what was discovered in biology when shoals of fish flock together as described by Hoare et al., (2000) or groups of birds (Newton, 2010). Swarm robotics can also enhance the robustness of the robots sent on hazardous assignments like mine clearance, chemical cleanup, military warfare etc., in which the failure of one or some of the robots does not mean the total failure of the mission but a gradual degradation of the system which can then be rectified by sending additional reinforcement. Information sharing is a vital process in algorithm design for flocking (Sumpter et al., 2008), each robot in the swarm disclose to other robots information about their positioning or their chosen motion bearing.

In swarm robotics, the problem of flocking entails directing a set of robots to move to a specific direction and converge to a target in an unfamiliar location. The robots that made up the swarm are required to accomplish this purpose as they are adjusting to their environments. Researchers have developed a number of control algorithms for flocking of swarm robots in the last few years. Reynolds (1987) was the one of the pioneers. He implemented a computer simulation to demonstrate the movement of a flock of birds, known as boids. This is based on the principal that the global behaviour of the boids as a whole is a direct outcome of the behaviours of every single participant (obeying certain instructions). There are three basic behavioural guidelines that every agent must have. They include:

separation, cohesion, and alignment. While separation will prevent the agents from bumping into one another, cohesion will make the agents to be united, and alignment will cause them to move with a collective speed.

In progression to Reynolds' work, e-boids was developed by Ward et al. (2001) to model the flocking behaviour of shoal of fishes. A down-to-earth flocking model of independent agents was developed by Vicsek et al. (1995), in which all the agents were assigned an unchanging fixed velocity. Jadbabie et al. (2003) went further to postulate hypothetical elucidations on the stated characteristics of the model presented by Vicsek. Some other researchers, Yang et al. (2008), did a similar work and proposed some rules for flocking in an anonymous environment with barriers in which the closest point on the barrier is considered a virtual agent.

The flocking behavior (in a distributed environment) based on artificial potential field (APF) was then investigated by Kim et al. (2006). They proposed a group of systematic procedures that can be used to create functions in the APF so that the agents in the swarm will not be trapped in local minima. The focus of all the aforementioned research work is based on collision avoidance among the robots without considering the flocking. Genetic programming was implemented by Spector et al. (2003) in a virtual environment to create collective behaviours for agents that hover in the swarm.

A robotic system that models the behaviour of physical ants that have the ability to locate the quickest route from one source of food to the subsequent one without any visible sign called pheromone was proposed by Payton et al. (2001). They also have the capability to quickly acclimatize to any environment variations that may require looking for a new quickest path if they encounter a new obstacle in the old one.

In this chapter, we proposed a hybridized swarm intelligence based algorithm, and a numerical optimisation algorithm i.e., a hybrid of Primal-Dual Interior-Point method and Particle Swarm Optimisation (PSO), to attain the best collective performance for considerable volume of swarm robots. The advantage of the novel hybrid Primal-Dual-PSO organization architecture is based in the truth that the flocking can be done in real time, it is decentralized, and inherently scalable because global communication is irrelevant in this matter. Decisions can be made by any of the agent using the local information available to them. The design is simple, scalable, adjustable, and has the ability to recover from catastrophic failure without disrupting its operations.

5.2.1 Problem Statement

In this study we assume that our swarm system consists of n entirely independent and identical (homogenous) robots. The robots in the swarm are individually designated by R_1, $R_2 \ldots R_n$ and they are represented as mobile points in two-dimensional space. The robots are given a neighborhood coordinate structure and they have limited capacity to perceive nearby robots. Moreover, the system is decentralized and there is no open interaction between the robots. The principal axis which defines the spatial locations of the environment is taken as the local axis of each of the robots. In our simulation, we used a point to represent each of the robot's two-dimensional space.

The velocity of the robot is updated at regular intervals and a maximum velocity is assigned to the robots. In addition, the robots flock in real time independently without any influence from other robots in the swarm. We are focusing on the convergence of the swarm to a specific point in the search space, and also the flocking patterns of the robots in the swarm. The Primal-Dual algorithm will calculate the positions, and afterwards, the PSO

algorithm will gain control of the flocking movement. The PSO algorithm will guide the flocking of the swarm (that comprises of N agents) to converge towards a global minimum and then flock around in the coordinate system. The flocking is done in real time during our simulation.

5.2.2 Using *pdPSO* to solve the flocking problem of swarm robotics

We applied the Primal-Dual-PSO algorithm to solving the flocking problem of swarm robotics. The Primal Dual algorithm will run until its tolerance is achieved on the objective function, and then the PSO phase is executed until its tolerance is achieved. From our experimental design, the search space is assumed to have a centre (c), then four (4) zones e.g. z1, z2, z3, and z4. For each experiment carried out, we want to know the number of iterations it takes the particles to converge to the centre (c), and the number of iterations it takes the particles to flock from c to z1, z2, z3, and z4. The different coordinates for different points that we used are as follows: Z1 (-30, 30), Z2 (-30, -30), Z3 (30, 30), Z4 (30,-30). The zones Z1, Z2, Z3 and Z4 are all points in the Cartesian coordinate plane. Our aim is that for each experiment carried out, we want to know the number of iterations it takes the particles to converge to any given point (p), and the number of iterations it takes the particles to flock from P to Z1, Z2, Z3, and Z4.

We used our Primal-Dual-PSO to make the particles to converge to the center point C(0,0) first, i.e., the particles will all be concentrated to a single point. Once they are there, we changed the target point to any zone that we want (Z1, Z2, Z3 or Z4) automatically in the script, and all the points that were converged to C(0,0), will start moving towards the zone in the form of a flock but will eventually converge again to Z1 in the form of a single point.

We set the termination and convergence criteria for the particles. When the objective function value is less than the tolerance, the algorithm will stop iterating. The threshold is not on gbest value, but on the value of the objective function. The Primal Dual brings down the objective value to 1 and then gives control to the PSO algorithm. Once, PSO brings down the objective function down to 1e-8, the algorithm stops iterating. The figure 5.1 below illustrates the search space and how the swarm flocks after converging at the centre from the centre to z1, z2, z3, and z4 respectively.

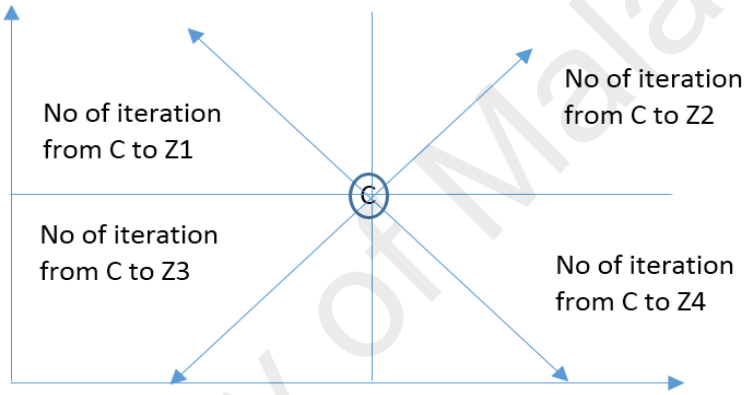


Figure 5.1: Convergence and Flocking strategy (The figure shows the zone 1, zone 2, zone 3, and zone 4 where the robots will flock to after converging at C)

We demonstrated the performance of our proposed hybrid Primal-Dual-PSO algorithm in a decentralized environment by simulating the movement of robots in the swarm in a virtual environment written in Matlab. There are some constraint that were set for the primal-dual algorithm based on the upper bound and the lower bound. The parameters for the PSO that were also set include the exploration environment which is a 200 x 200 dimension coordinate system. Each of the robots is symbolized with the black dots, the system is designed to operate in real time. The number of robot and the dimension of the search is specified in the GUI. This simulation was run on a MATLAB R2013a on an Intel ® Core ™

i3-2328M machine with 4GB memory running Windows 7. We designed a GUI to simulate how the robots converge to a point and then start flocking by simply clicking the mouse at the point where we want them to converge. The GUI is user-friendly and it easy to change the parameters. The user can halt, suspension, or restart the simulation any time he feels like.

5.2.3 Result and Discussion

At the beginning of simulation, there is a haphazard allocation of agents A^i ($0<i\leq N$) in the environment to be explored. The new algorithm works by first starting the agents' positions randomly. Then, the agents are directed to the Primal-Dual method, which gives us its initial optimisation result after some number of iterations. The result of the Primal-Dual optimisation is then feed into PSO, which creates a perturbation in the population and also maintain diversity in the population until there is either convergence to the global optimal or the termination criteria is reached.

During the iterations that are performed in the optimisation, the agents carry out shift in the search space from one point to the other (which have connection to behaviour evolutions in space). The cognitive and social scaling factors of the PSO are very important parameters for determining the optimal global behaviors of the agents in the swarm. Figures 6.2 – 6.7 display how the robots converged to the centre and how they flocked from the centre to different zones using the *pdPSO* algorithm.

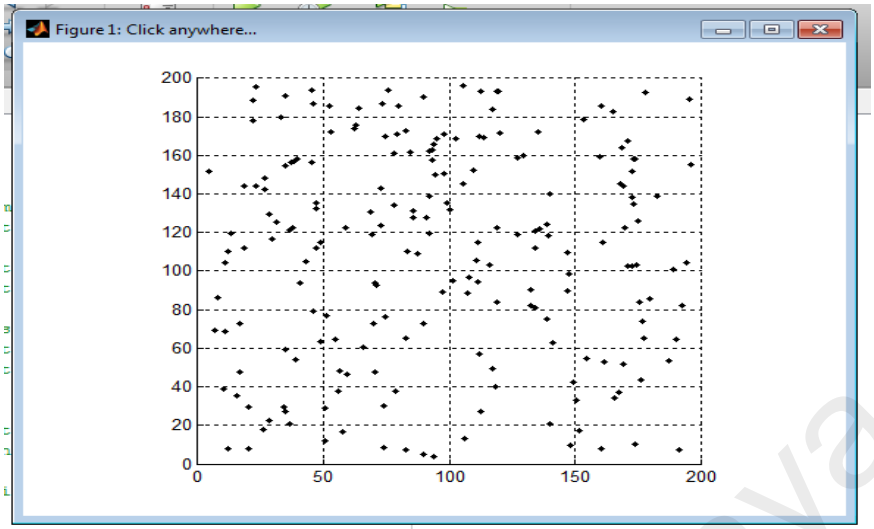


Figure 5.2: Screenshot of 200 robots (The black dotted lines represent the robots that are in the search space. The *pdPSO* algorithm is used to control the movement of the robots.)

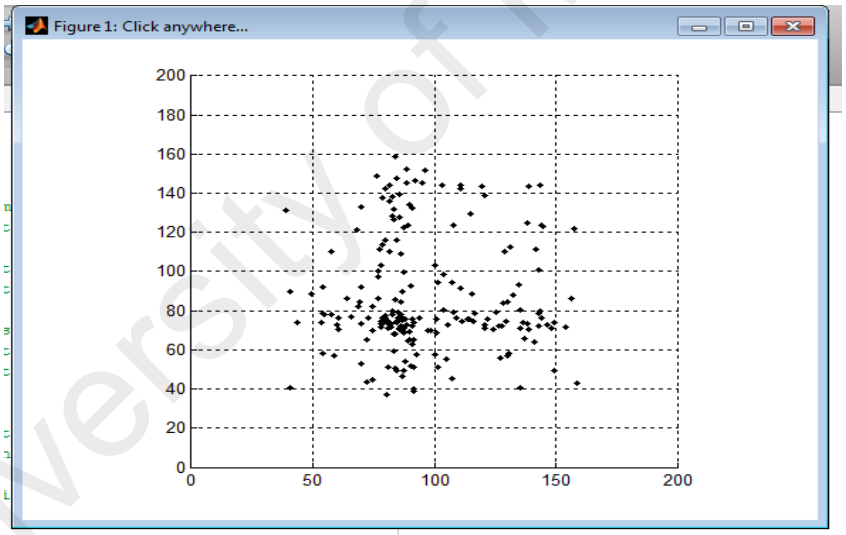


Figure 5.3: Screenshot robots moving towards convergence (The 200 robots moves towards the convergence point).

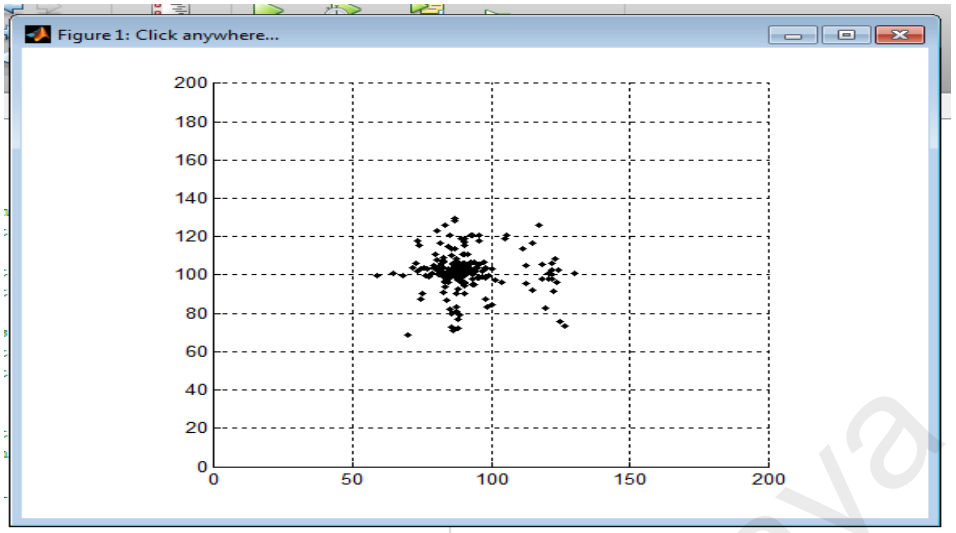


Figure 5.4: Screenshot robots converging at a point (The black dotted lines represent the 200 robots as they converge to the centre).

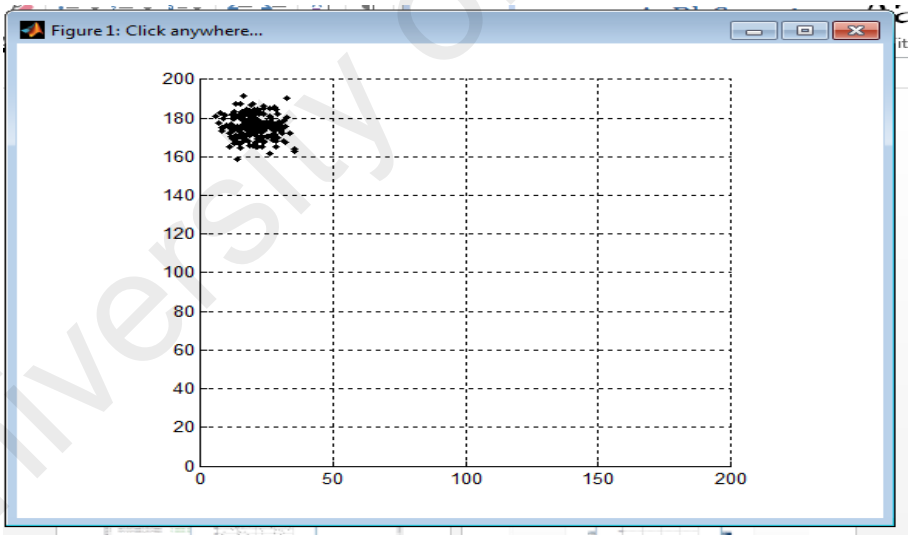


Figure 5.5: Screenshot robots flocking to zone 1 (The black dotted lines represent the 200 robots as they flock from the centre to zone 1).

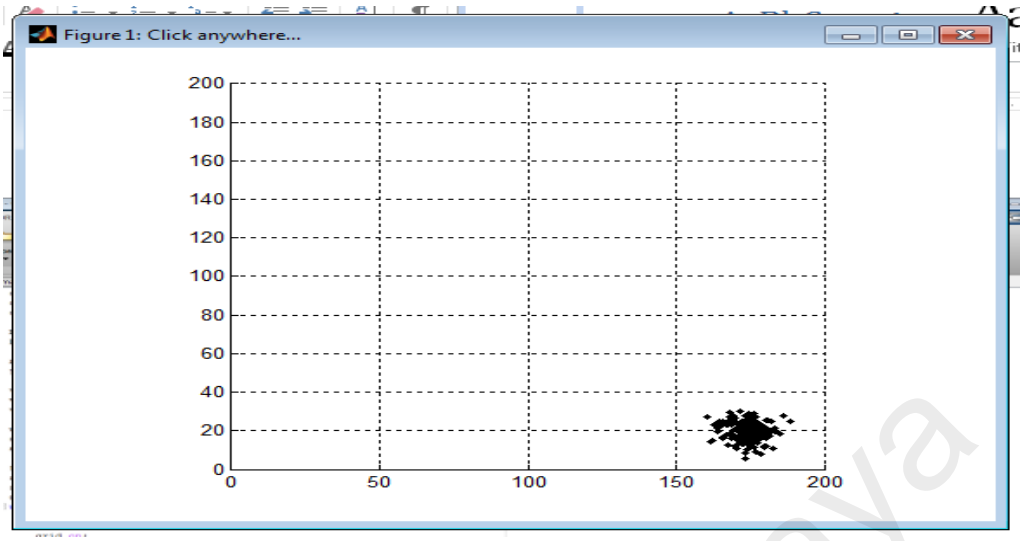


Figure 5.6: Screenshot robots flocking to zone 4 (The black dotted lines represent the 200 robots as they flock from the centre to zone 4).

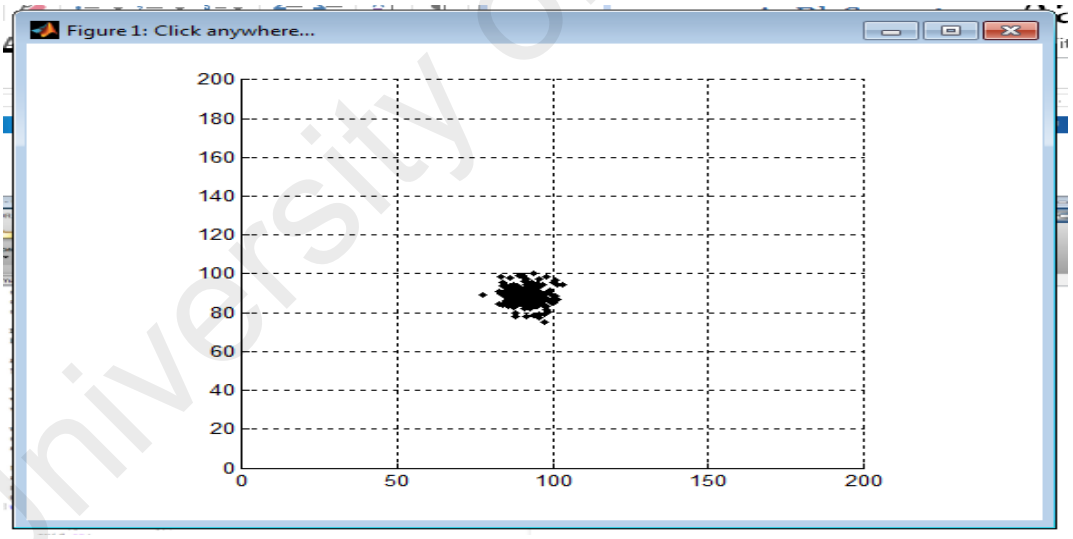


Figure 5.7: Screenshot robots flocking to zone 3 (The black dotted lines represent the 200 robots as they flock from the centre to zone 3).

For *pdPSO* we carried out some simulations using 100 particles (robots) with each particle having a dimension of 10. The PSO tolerance is 1.000000e-08, and IPM Tolerance is 1.000000e+00. The result of our simulation for the robots flocking to Zone 1 having coordinate points (-40, 40) is presented in Appendix C. The result of our simulation of the robots flocking to Zone 2 is presented in Appendix D. Appendix E contains the simulation result of the robots flocking to Zone 3. And the simulation result of the robots flocking to Zone 4 is presented in Appendix F. We summarized the values of mean and the variance of the number of iteration it took to converge to the centre and flock to the target zone in the table 5.1 below.

Table 5.1: Mean and Variance for Convergence and flocking using *Primal-Dual-PSO*

Zone	Activity	Mean	Variance
Z1	Convergence	197.820	337.906
	Flocking	335.060	428.915
Z2	Convergence	195.880	174.026
	Flocking	340.600	318.857
Z3	Convergence	195.160	203.811
	Flocking	325.920	299.381
Z4	Convergence	194	261.909
	Flocking	329.320	311.161

The graphs of the total iteration to converge at the centre C and that of flocking to the different zones for all the simulations we carried out are below.

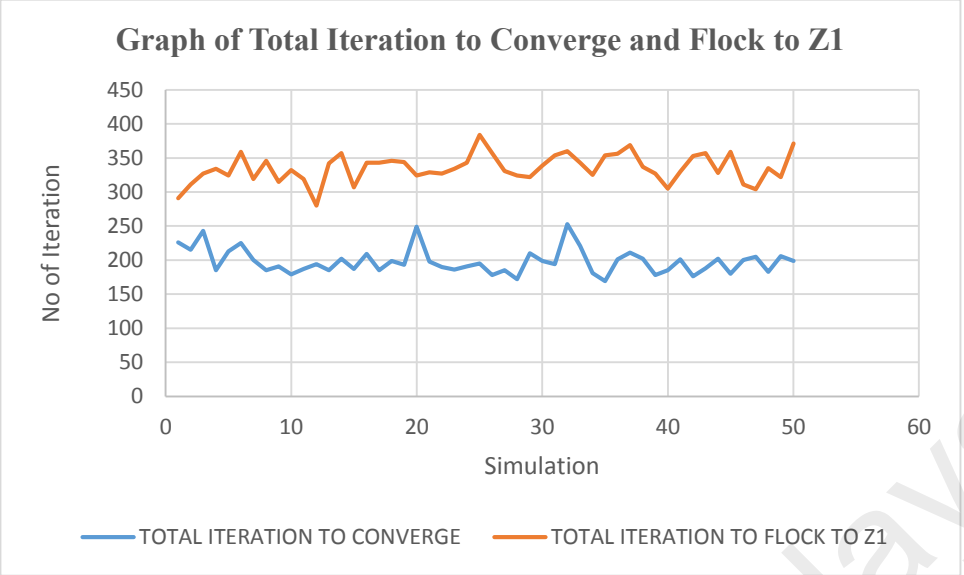


Figure 5.8: Graph of total iteration to converge and flock to Zone 1

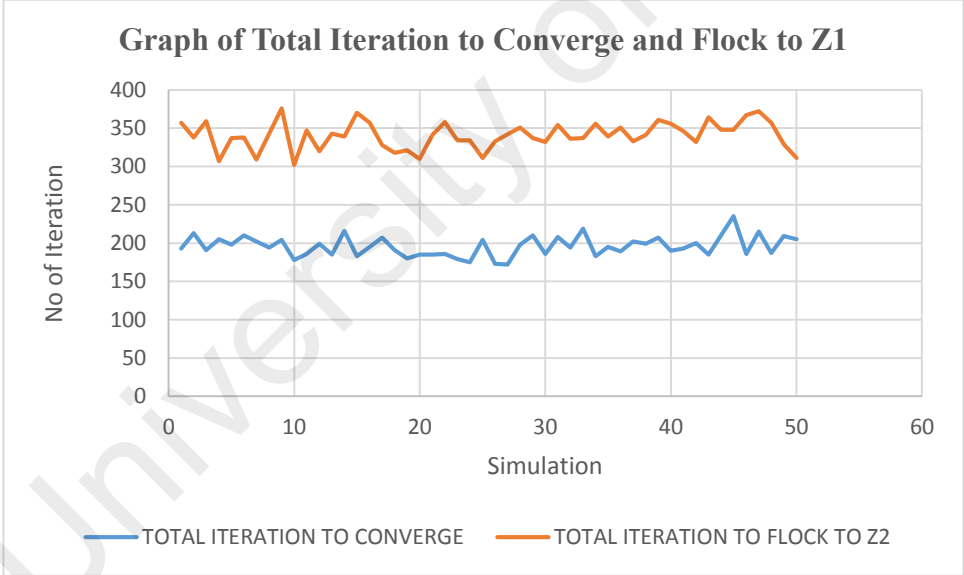


Figure 5.9: Graph of total iteration to converge and flock to Zone 2

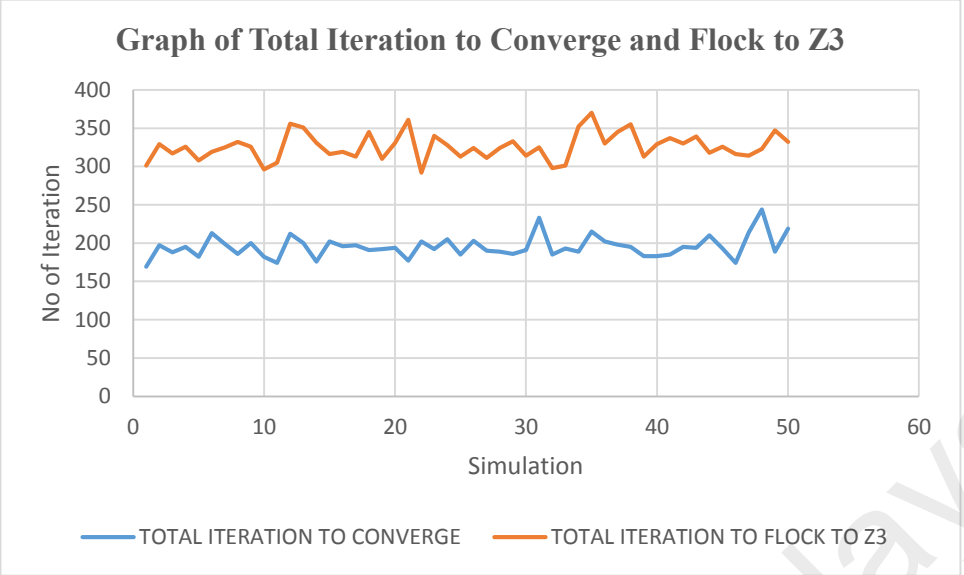


Figure 5.10: Graph of total iteration to converge and flock to Zone 3

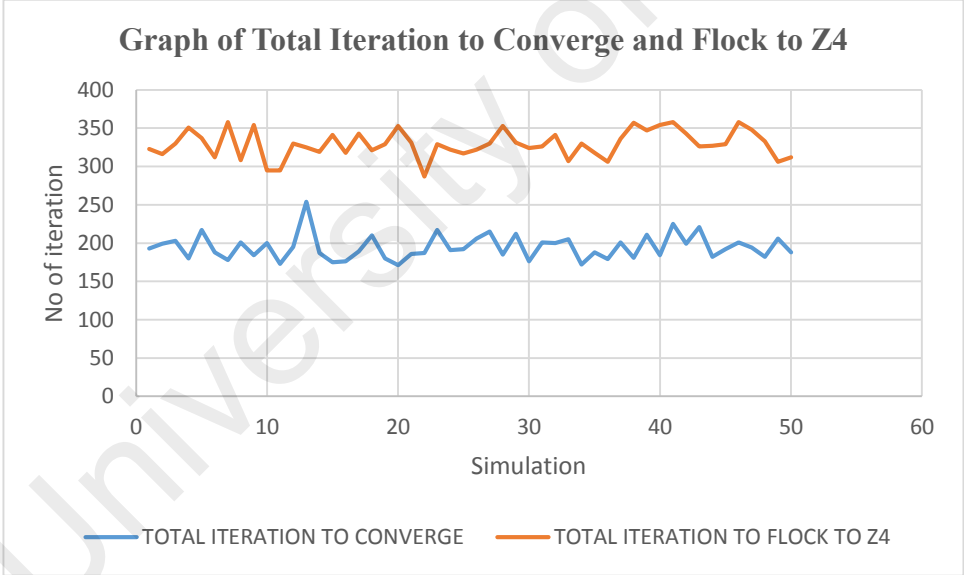


Figure 5.11: Graph of total iteration to converge and flock to Zone 4

Figures 5.12 – 5.17 below display how the robots converged to the centre and how they flocked from the centre to different zones using the *pdAPSO* algorithm.

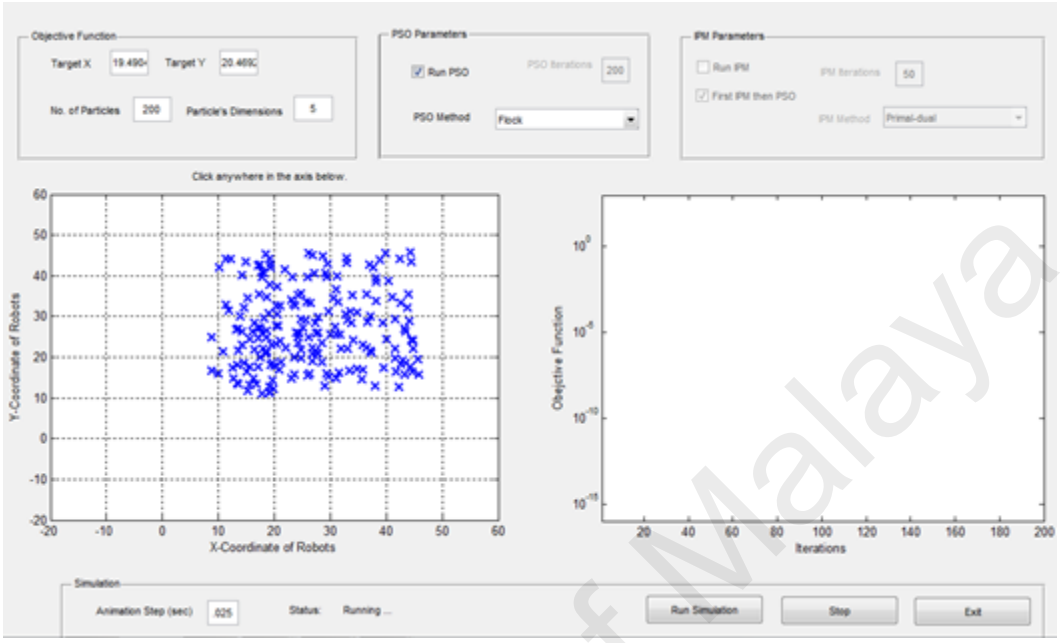


Figure 5.12: Screenshot of robots moving towards the centre (the blue dots in the figure depicts the 200 robots as they move towards converging at the center as shown in the GUI.)

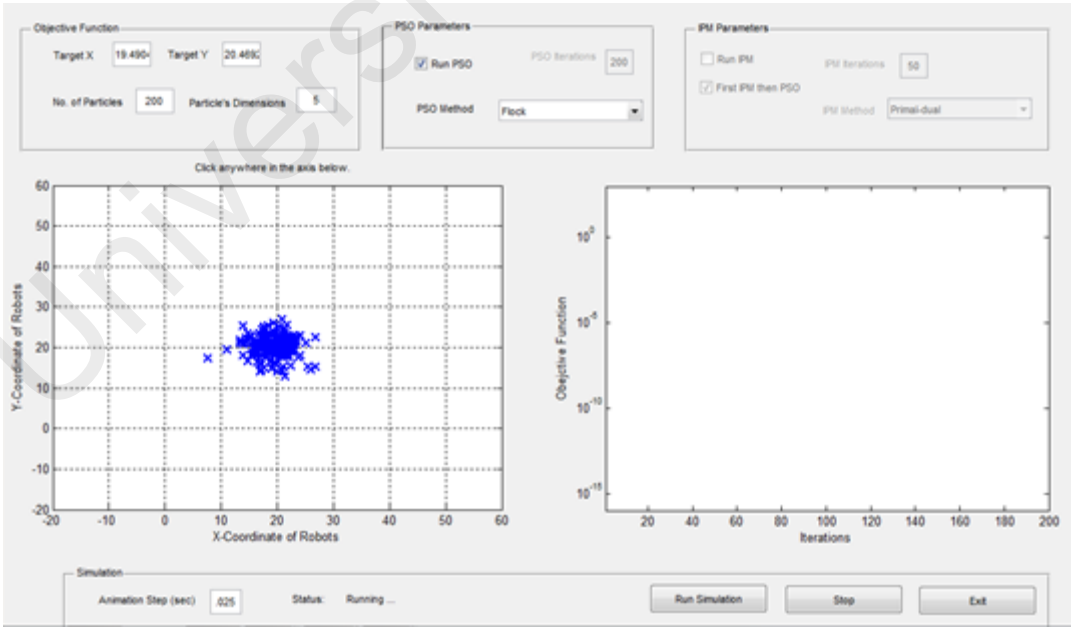


Figure 5.13: Screenshot of robots converging at the centre (the blue dots depicts the robots converging at the centre.)

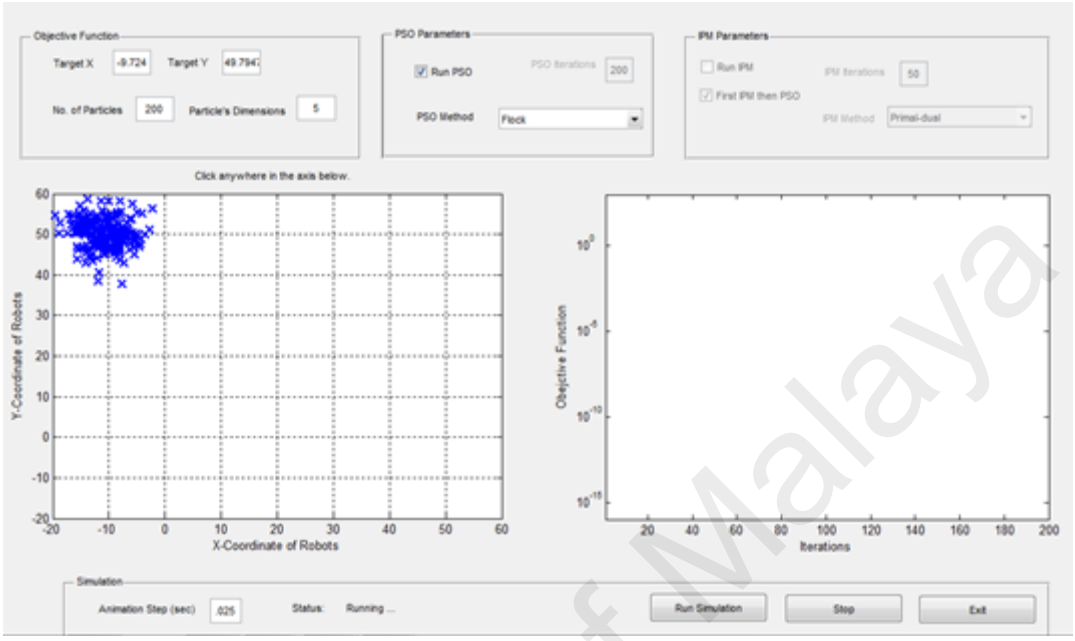


Figure 5.14: Screenshot of robots flocking to zone 1 (the blue dots in the figure depicts the robots flocking from the centre to zone 1).

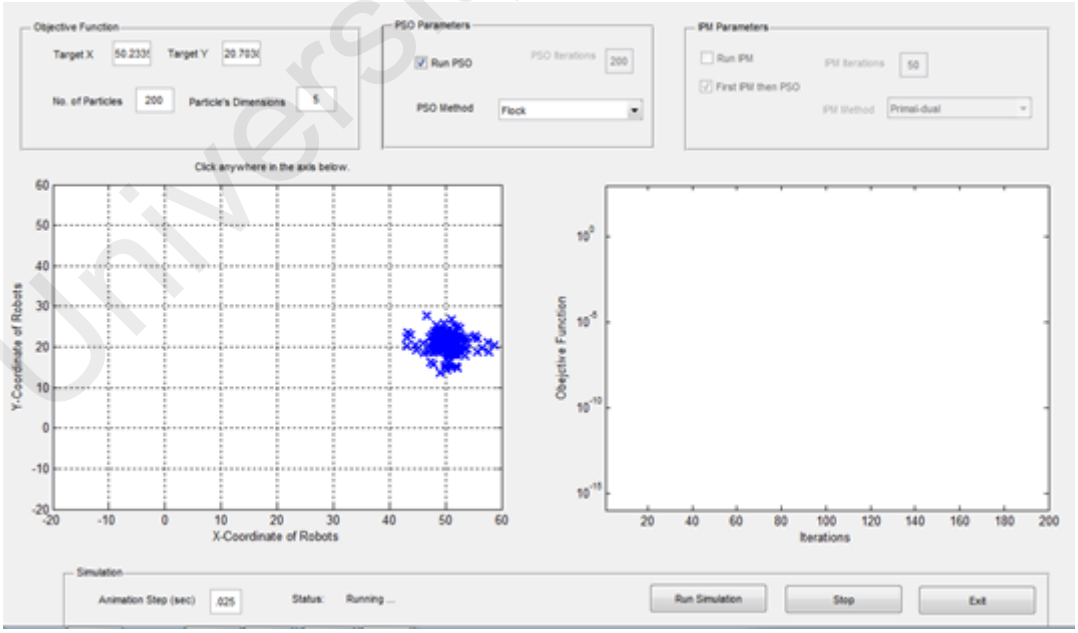


Figure 5.15: Screenshot of robots flocking to zone 2 (the blue dots in the figure depicts the robots flocking from the centre to zone 2).

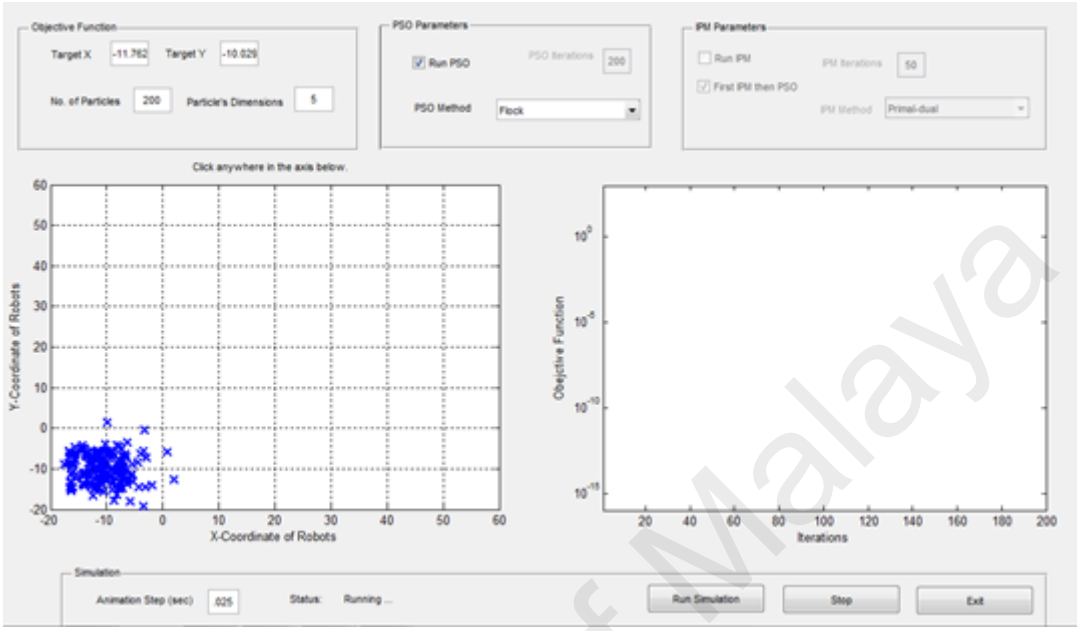


Figure 5.16: Screenshot of robots flocking to zone 3 (the blue dots in the figure depicts the robots flocking from the centre to zone 3).

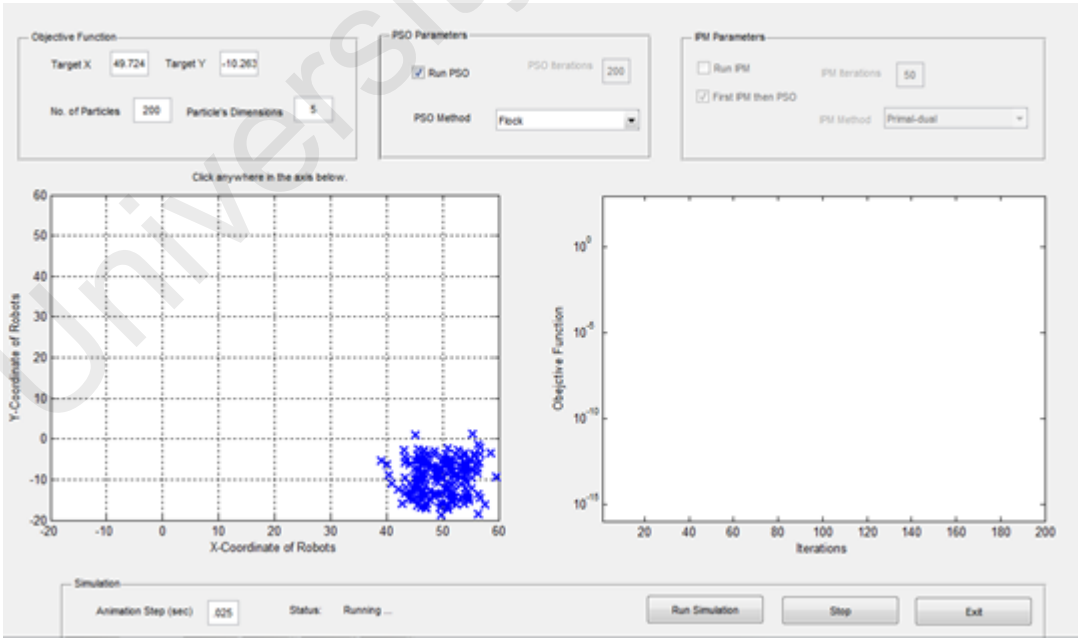


Figure 5.17: Screenshot of robots flocking to zone 4 (the blue dots in the figure depicts the robots flocking from the centre to zone 4).

We did some simulations for our *pdAPSO* algorithm. Using 100 particles (robots) with each particle having a dimension of 10. The APSO tolerance is 1.000000e-08, and IPM Tolerance is 1.000000e+00. The result of our simulation for the robots flocking to Zone 1 having coordinate points (-40, 40) is presented in Appendix G. The result of our simulation of the robots flocking to Zone 2 is presented in Appendix H. Also presented is the simulation result of the robots flocking to Zone 3 as shown in Appendix I. And the simulation result of the robots flocking to Zone 4 is presented in Appendix J. We summarized the values of mean and the variance of the number of iteration it took to converge to the centre and flock to the target zone (Z1, Z2, Z3, and Z4) in the table 5.2 below.

Table 5.2: Mean and Variance for Convergence and flocking using *Primal-Dual-APSO*

Zone	Activity	Mean	Variance
Z1	Convergence	209.62	260.44
	Flocking	149.547	569.966
Z2	Convergence	211.28	159.063
	Flocking	259.52	472.5
Z3	Convergence	210.34	187.045
	Flocking	259.22	484.053
Z4	Convergence	1030.440	2.088
	Flocking	1009.0	0.00

To evaluate the effectiveness of flocking capability of our *pdAPSO* algorithm we compared its performance with other existing algorithms such as the PSO, APSO and Primal Dual. We observed from our simulations that for Primal Dual algorithm, the robots in the swarm flock very tightly as a single point. When the agents in the swarm move from random starting position to converge at the centre (0, 0), they become almost one point, which means they are tightly flocked. When they move from the centre position (0, 0) to different zones,

they are so tightly flocked that they move almost as a single point. The result is that no matter the zone, it always takes equal number of iterations to reach different zones (10 as given in results from the table in the supplementary materials). Therefore, we can say that flocking in Primal Dual is very tight compared to what is obtainable in *pdAPSO*, APSO and PSO in which particles still move around a center point randomly. The issue of tightness of the robots in the swarm when using Primal Dual for flocking is impracticable in the real world sense as a safe distance must be maintained between the robots to avoid collision.

Though Primal Dual have the lowest number of iterations to flock from one zone to another zone followed by *pdAPSO*, APSO and PSO respectively. We however observed from our results that the value for number of iterations to flock from one zone to another for Primal Dual is constant (value 10). This is understandable because in Primal Dual-based methods, there is no random number involved. Therefore, whether the robots move from the centre (0, 0) to z1, z2, z3 or z4, the algorithm moves in a deterministic way. As long as the distance between the centre (0, 0) and z1, z2, z3 and z4 points are equal, Primal Dual takes equal number of iterations. The fact that the number of iterations is constant further confirms the inability of the particles in Primal Dual algorithm to escape being trapped in the local minimal. We posit that Primal Dual having a lower number of iterations is not an index that it is better than any of the other three (3) algorithms we compared in this paper. It is possible that one iteration of Primal Dual is more complex and has more computations than one iteration of *pdAPSO*, APSO and PSO. However, there is a lot of variation in *pdAPSO*, APSO and PSO iterations when the robots in the swarm move from the centre (0, 0) to different zones, whereas in Primal Dual the variance is very low. The lower variance shows that Primal Dual keeps the particles tightly knit together compared to *pdAPSO*, APSO and PSO. If you look at the formula for PSO-based algorithms (*pdAPSO*, APSO and PSO), it has random

numbers involved. Therefore, when we run the *pdAPSO*, APSO and PSO methods, every time the algorithm has slightly different trajectories for the robots. The different trajectory results in different number of iterations every time.

In appendix J, we have the statistical results of the experiments that we performed. The numerical values of the total number of iterations for the Primal-Dual-APSO *(pdPSO)* to get the robots to converge at the centre (C) are in the appendix. Also, the total number of iteration for the robots to flock from the centre to zone 1 is presented in the appendix. The computed values of the mean and variance is also included. The values of the number of iterations for the *pdAPSO* to get the robots to converge at the centre (C) is in the table. Also, the total number of iterations for the robots to flock from the centre to zone 2 is presented. The computed values of the mean and variance is also included. We also have the results of our experiments for flocking from the centre to zone 3. The numerical values of the total number of iterations for the *pdAPSO* to get the robots to converge at the centre (C) is in the table. Also, the total number of iterations for the robots to flock from the centre to zone 3 is presented. The computed values of the mean and variance are also included. The appendix M also shows the results of our experiments for flocking from the centre to zone 4. The numerical values of the number of iterations for the *pdAPSO* to get the robots to converge at the centre (C) is in the appendix. Also, the total number of iteration for the robots to flock from the centre to zone 4 is presented. The computed values of the mean and variance is also included. Table 5.2 presents a summary of the mean and variance of the total number of iterations to converge to the centre and to flock from the centre to the different zones (zone 1, zone 2, zone 3, and zone 4). Figures 5.18 – 5.21 show the graph of total iteration to converge to the centre and flock from there to the different zones (zone 1, zone 2, zone 3, and zone 4) using *pdAPSO*.

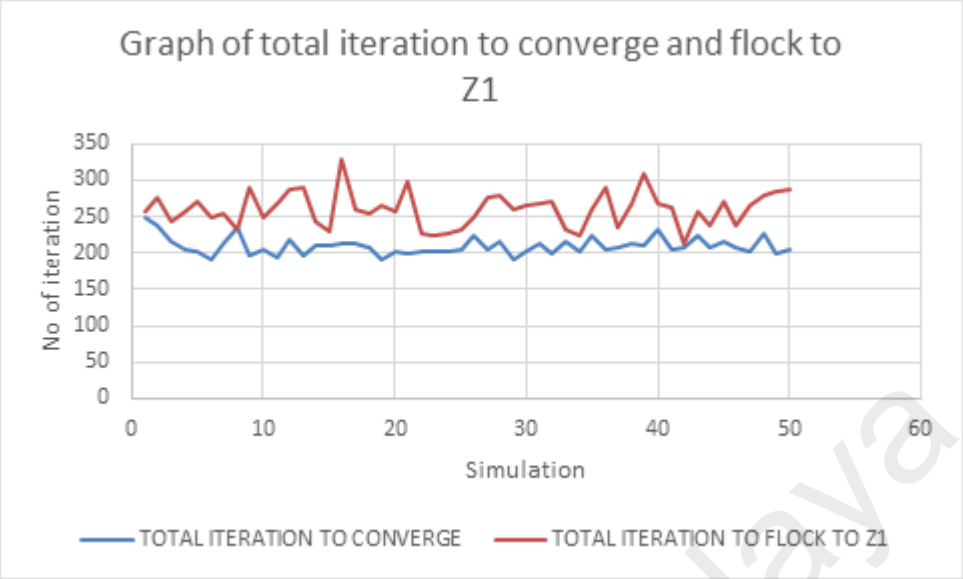


Figure 5.18: Graph of total iteration to converge and flock to Zone 1. This is a graphical illustration of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone one (1) of the search space using the *pdAPSO* algorithm.

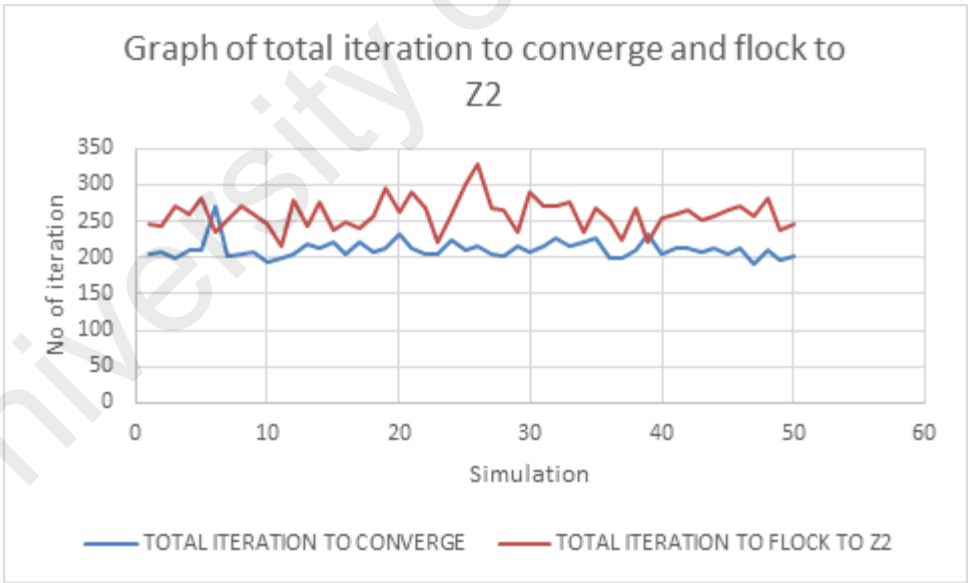


Figure 5.19: Graph of total iteration to converge and flock to Zone 2. The above figure depicts the graphical representation of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone two (2) of the search space using the *pdAPSO* algorithm.

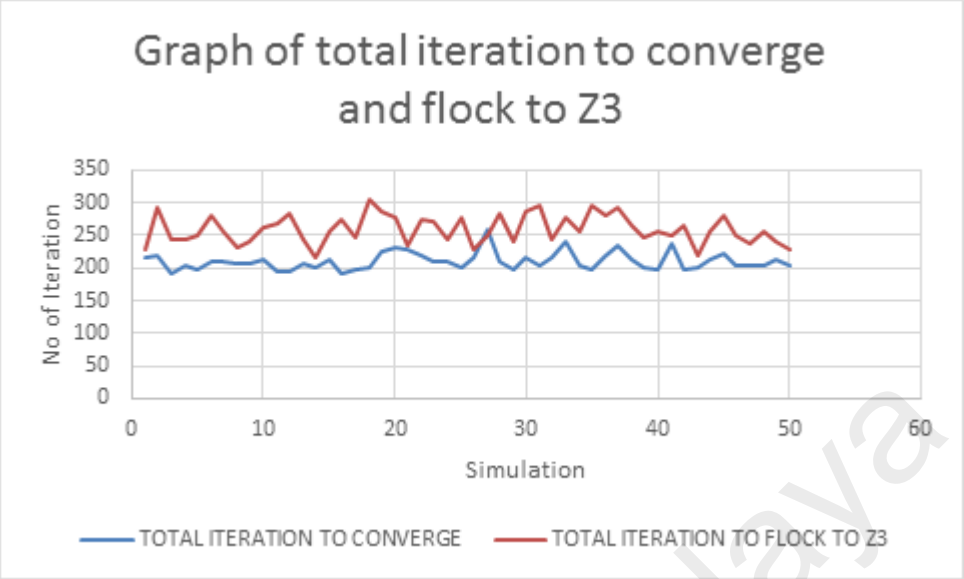


Figure 5.20: Graph of total iteration to converge and flock to Zone 3. Above is a graph of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone three (3) of the search space using the *pdAPSO* algorithm.

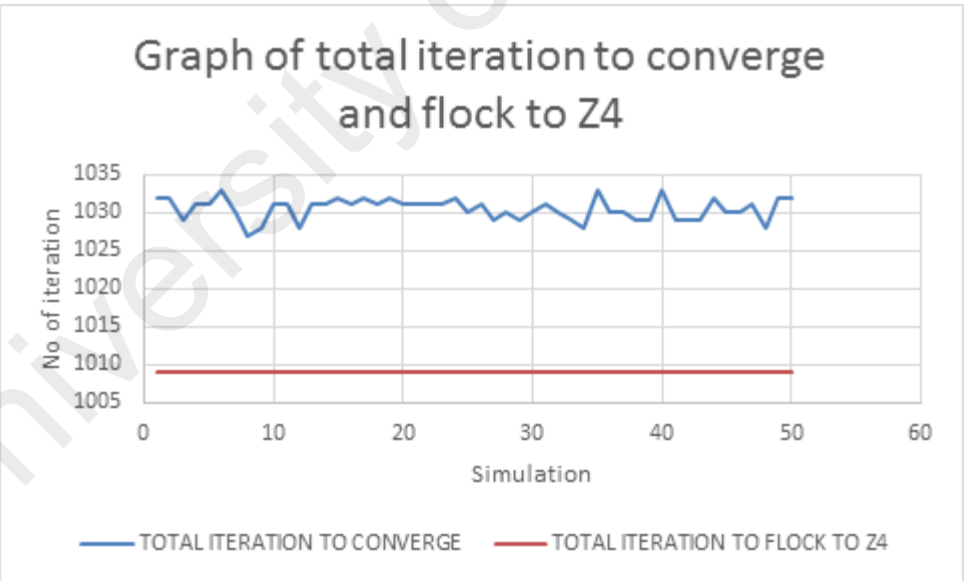


Figure 5.21: Graph of total iteration to converge and flock to Zone 4. The figure above shows the graphical depiction of the number of iterations it took the robots to converge to the center (0, 0) and flock to zone four (4) of the search space using the *pdAPSO* algorithm.

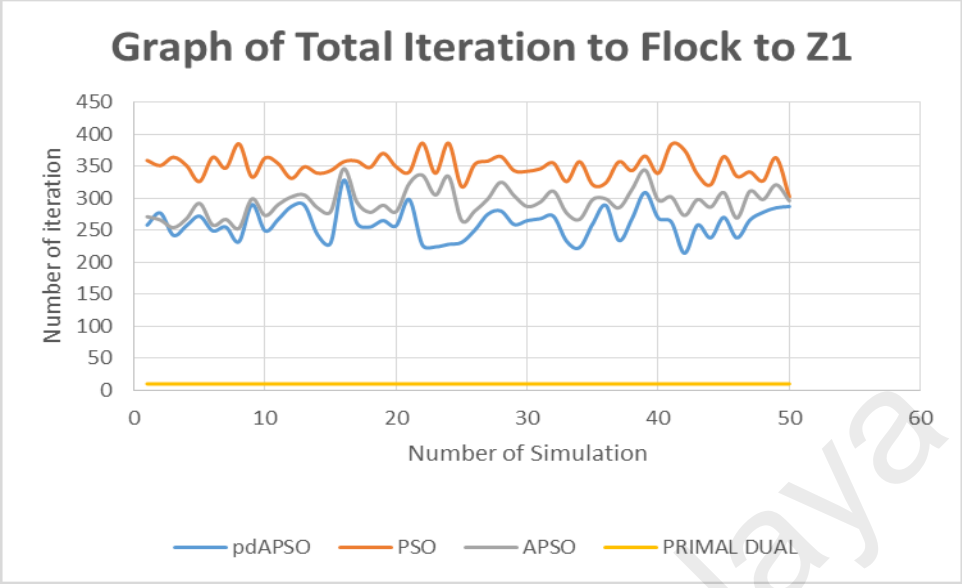


Figure 5.22: Graph of performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 1. The *pdAPSO* performs better than the other three (3) algorithms by having the lowest number of iterations in the fifty (50) simulation that was done (except for Primal Dual that have a constant number of iterations for each of the simulations because of its deterministic nature).

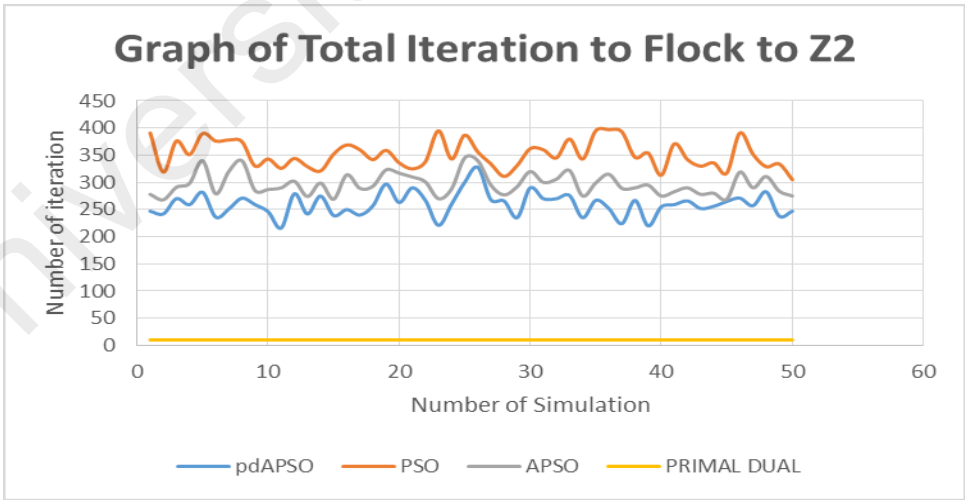


Figure 5.23: Graph of performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 2. The performance of *pdAPSO* is better than the one of the other three (3) algorithms by its ability to flock to

zone 2 in using the minimum number of iterations in the fifty (50) simulation that was done (except for Primal Dual that have a constant number of iterations for each of the simulations because of its not a heuristic algorithm).

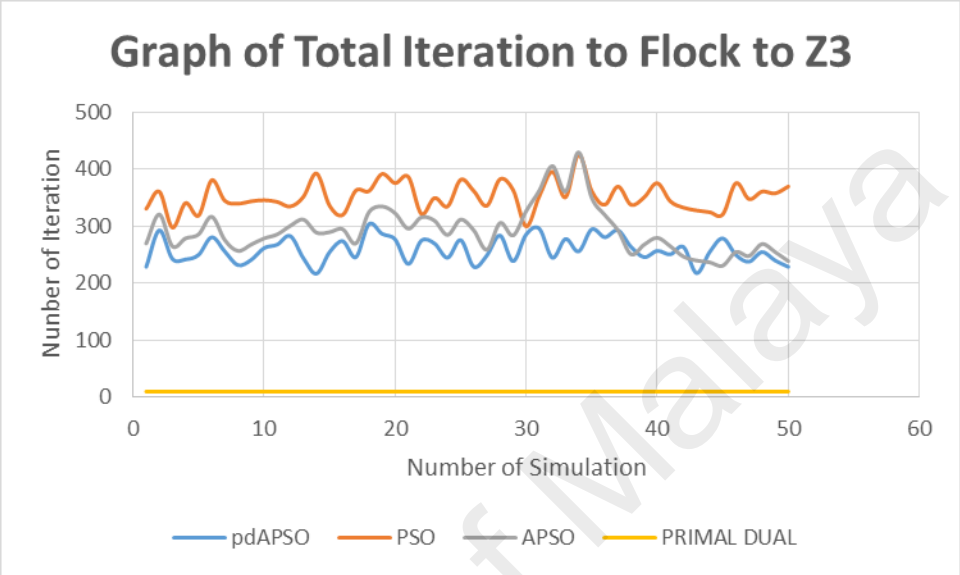


Figure 5.24: Graph showing the performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 3. Our algorithm *pdAPSO* shows a better flocking capability than that of PSO, APSO and Primal Dual algorithms by allowing the robots to flock to zone 3 by having the minimum number of iterations in the fifty (50) simulations that was done (except for Primal Dual that have a constant number of iterations value 10 for each of the simulations since it is a heuristic algorithm).

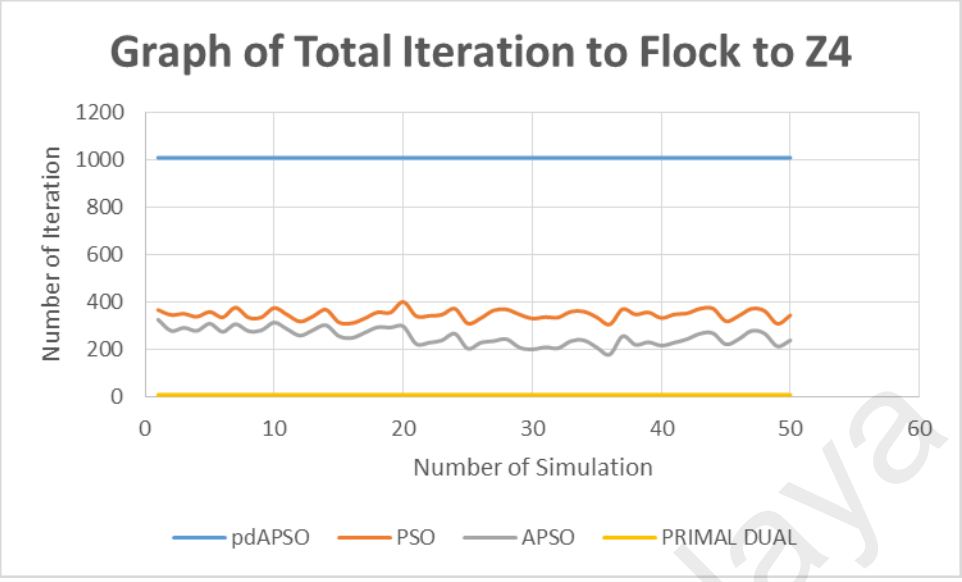


Figure 5.25: Graph showing the performance comparison of total iteration for *pdAPSO*, PSO, APSO and Primal Dual algorithms as the robots flock from the centre to Zone 4. The APSO algorithm performs better than *pdAPSO,* PSO and Primal Dual algorithms in this scenario by having the minimum number of iterations in the fifty (50) simulations that was done. The *pdAPSO* and Primal Dual a constant number of iterations value 1009 and 10 respectively for each of the simulations.

Upon closer inspection, the proposed algorithm (*pdAPSO*) performed better compared to its predecessors in zone 1, zone 2 and zone 3. Across the 50 simulations that were executed for each zone, *pdAPSO* consistently generates the lowest number of iteration of flocking the swarm to the desired location. Table 5.3 summarised the mean iterations of each algorithm for each zone. Interestingly, the proposed algorithm performed flat in zone 4. The algorithm converged to the maximum iteration number, indicating that the algorithm was trapped in local minima. This behaviour is equivalent to the deterministic traits of the primal dual algorithm. It is therefore important to investigate this anomaly to improve on the robustness aspect of the proposed algorithm.

Table 5.3: The mean number of iterations for *pdAPSO*, PSO, APSO and Primal dual algorithms for all runs.

Flocking converges to the centre than to each zone. Individual numbers are available in supplementary material.

Algorithm	Centre	Zone 1	Zone 2	Zone 3	Zone 4
pdAPSO	209.62±14.41944	260.44±14.72325	259.52±14.43984	259.22±14.39885	1009.00±0
PSO	288.56±21.8175	349.10±23.70584	351.64±21.83863	352.48±21.78671	348.18±0
APSO	275.82±17.89710	293.10±18.69183	296.78±21.90125	292.80±22.28676	254.84±17.1977
Primal Dual	10.00±0	10.00±0	10.00±0	10.00±0	10.00±0

5.3 Pattern Formation

Pattern formation basically involves forming a particular shape by altering the positions of each robots in the swarm. The most common problem in the field of swarm robotics deals with managing and directing how a sizable number of robots navigate in the search environment to accomplish an assignment in unity. This type of task is normally impracticable, demanding and laborious for a particular set of robots to accomplish. The inspiration of swarm robots is fundamentally drawn from the study of behaviour of animals like the flock of birds, herd of cattle, and shoal of fish. The performance of the swarm at the global level will be largely influenced by the performance of the individual agent at the local level. Some of the traits of swarm robotics that have been extensively investigated are convergence, foraging (Krieger, Billeter & Keller, 2000), pattern formation (Balch & Arkin, 1999), flocking, aggregation and segregation (Martinoli, Ijspeert, & Mondada, 1999), box-pushing (Mataric, Nilsson & Simsarian, 1995), cooperative mapping (Yamauchi, 1999),

soccer tournaments (Weigel, 2002), site preparation (Parker & Zhang, 2006) and sorting (Holland, & Melhuish, 1999).

Researchers of recent have shifted much of their attention to the do extensive study on multi-robot systems from the perspective of engineering and artificial intelligence. Pattern formation have been considered as an initial phase for an effective flocking for many reasons which comprise of corresponding carriage of loads, preventing intrusion etc.) The aim of the pattern formation problem is to build some guidelines and principles that will make the robots in the swarm to work as a group from the local level to achieve complex overall tasks

Multi-robot pattern formation is an extremely needed solution in majority of the problem areas where a swarm of robots are employed and it is essential to organize them in a particular manner. The formation can be described as an arrangement in a constrained working area, in which individual robot is given a predefined gap between them and their neighbours. Multi-robot pattern formation is defined as a configuration in a bounded workspace, where each robot is at a desired distance from its neighbors. The desired formation is specified in terms of relative distances, so that the formation can be achieved in any part of the workspace and at any orientation.

Our aim is to present a solution to the problem of pattern formation on a grid map, for a homogeneous multi-robot system using Primal-Dual Particle Swarm Optimisation (*pdPSO*) model and Virtual Pheromone mechanism. Basically, a virtual pheromone trail based method is proposed as the message passing mechanism among the robots, where robots make distributed movement decisions through local interactions. For one individual robot, there are two working modes, exploration and dispersion, with different indicators in the *pdPSO* model. By cooperating and communicating through the virtual pheromone, agents of the multi-robot system switch between the two working modes. The *pdPSO* method helps to

allocate reasonable robots to different parts of the predefined pattern. A series of experiments was carried out and proves the convergence and excellent scalability of our algorithm. By optimizing some parameter in the *pdPSO* model, the efficiency of pattern formation is further improved.

The virtual pheromone was introduced to guarantee the effectiveness of harmonization among the robots and to prevent the robots from gathering in a certain part of the pattern and being totally absent in some other parts. For the start, the initial value of the pheromone is set to 0. The robot mimic the natural boosting action of pheromone in biology by updating the pheromone and propagating the information to its neighbours that are within the limited communication range anytime it locates a grid that is still available for occupation. The maximum value of pheromone is set to prevent run-off. The virtual pheromone level will continue to decrease with time until it is finally eradicated. Figure 5.30 below is a pictorial representation of how the pheromone works.

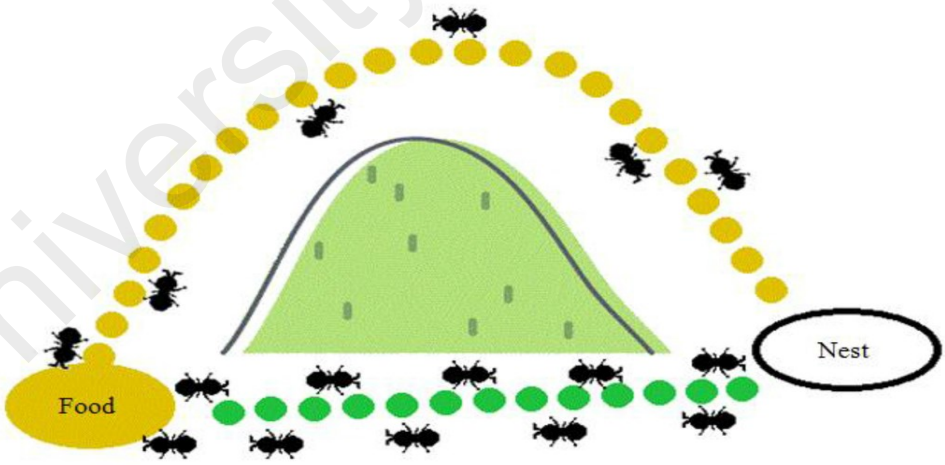


Figure 5.26: Pictorial Representation of Pheromone

5.3.1 World Definition

We presume that the environment consists of an M×N grid-based map. We vary the number of agents distributed in the map to 50, 100, 150, 200, and 250. Our method uses one type of pheromone and also holds the map of pattern (location of the grids of pattern found by the current agent or agents near to the current agent) in its memory. The pheromone is used for exploration (searching the map) and the map of pattern is used for convergence (converging to the grids of pattern found). The pheromone is updated when an agent moves in the map and the map of pattern is updated when an agent found a grid of pattern. However, agents communicate with other agents nearby and try to update pheromone and map of the pattern. In exploration mode, an agent moves in a deterministic manner based on the pheromone (moving to the grids with lower pheromone to explore unvisited grids). In convergence mode, agent moves based on map of pattern semi-stochastically (moving to the grids with higher pheromone to approach to the grids of the pattern). In this method, we use sub-area (blocks) for pheromone rather than sub-area (blocks) for map of pattern unlike the approach used in the work of Xu et al. (2010). This means that the agent must hold a matrix of pheromone, and the size of this matrix must be smaller than that of the map. It must also hold a matrix for map of pattern so that the size of the matrix is equal to the size of the map. Some of the presumptions that guide the work of the agents of this swarm robotic system are as follow:

(1) Every agent in the system is indistinctive and similar; meaning they cannot be differentiated by their exterior look;

(2) There is a limited range of communication for individual agents;

(3) There is an accurate recording of the decentralization and navigation of agents in the grids;

(4) Agent to agent, or agent to obstruction collisions are insignificant.

In figure 5.31 below, the flowchart for the operation of pheromone is presented.

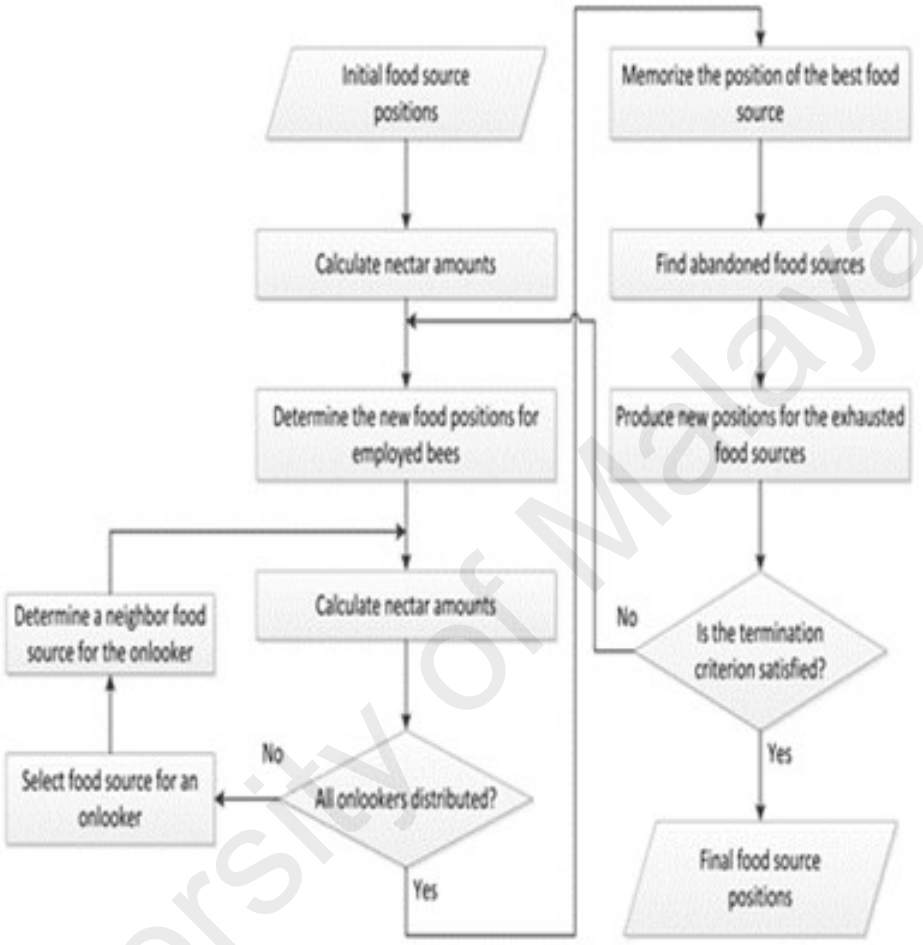


Figure 5.27: Pheromone Flowchart

Figure 5.32 below is the pictorial representation of the virtual pheromone grid map.

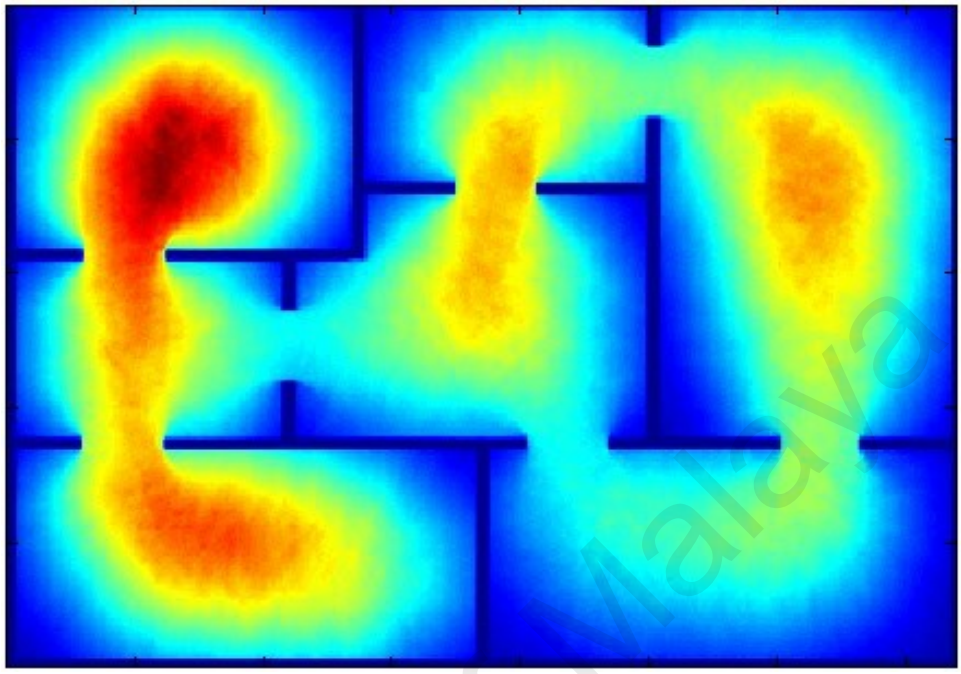


Figure 5.28: Virtual Pheromone Map

5.3.2 Problem Statement

Assuming that we have an ellipse pattern on the grid of map, which is a group of coordinates. The pattern formation problem is the problem of designing an algorithm that is decentralized, in a way that, as the agents are distributed randomly, they will finally form the coveted ellipse pattern. Here, we have an unfamiliar area and a pattern defined beforehand. The swarm which is initially started randomly, searches grids for the coordinates that represent the pattern and ultimately complete taking the shape of the preferred pattern The aim of our experiment is to build an ellipse pattern formation algorithm that can allow more agents to be added to the number of agents in the swarm and thereby allowing the swarm robot system within the least time frame to form the predefined pattern.

5.3.3 Pattern Formation Algorithm

At the onset, all the robots are haphazardly positioned on the map, and some of the grids on the map are designated as the predefined pattern where the information are stored in the matrix. A local search is first conducted since the robots are positioned randomly on the map with the robots set to the spreading activated mode searching for an unpopulated designated grid. With the spreading mode activated, each agent uses the virtual pheromone to communicate the information of its own local area in the grid with its neighbours, thereby increasing learning. A Primal-Dual PSO (*pdPSO*) based searching approach that uses pheromone information is used for allotting local jobs among robots. The algorithm uses the value of distance that exists amid grids as a guide to update the location of the agent on the map. The exploration mode is activated whenever the agent notices that there is no vacant designated grid in the sub-area, enabling it to find a nearby occupied sub-area that have a limited number robots now occupying it. This operation continues, pending the time when all the robots sent to the designated grids and the pattern specified by the designated grids is created by the robots in the swarm. We invented two approaches for switching between exploration and convergence:

1 A certain number of iterations (e.g. 80) for exploration and then a certain number iterations (e.g. 20) for convergence. Of course, the iterations for exploration are always performed before the iterations for convergence.

2 Stochastic switching between exploration and convergence. In this approach, the probability of exploration mode in the first iteration is very high and gradually, the probability of convergence mode is increased. In the last iterations, the probability of convergence mode is very high. We combine the two approaches mentioned above. There is stochastic switching between exploration and convergence while the probability of exploration mode is zero if the number of iterations is more than the

threshold. In this case, the probability of convergence mode is very low when the number of iterations is less than the threshold.

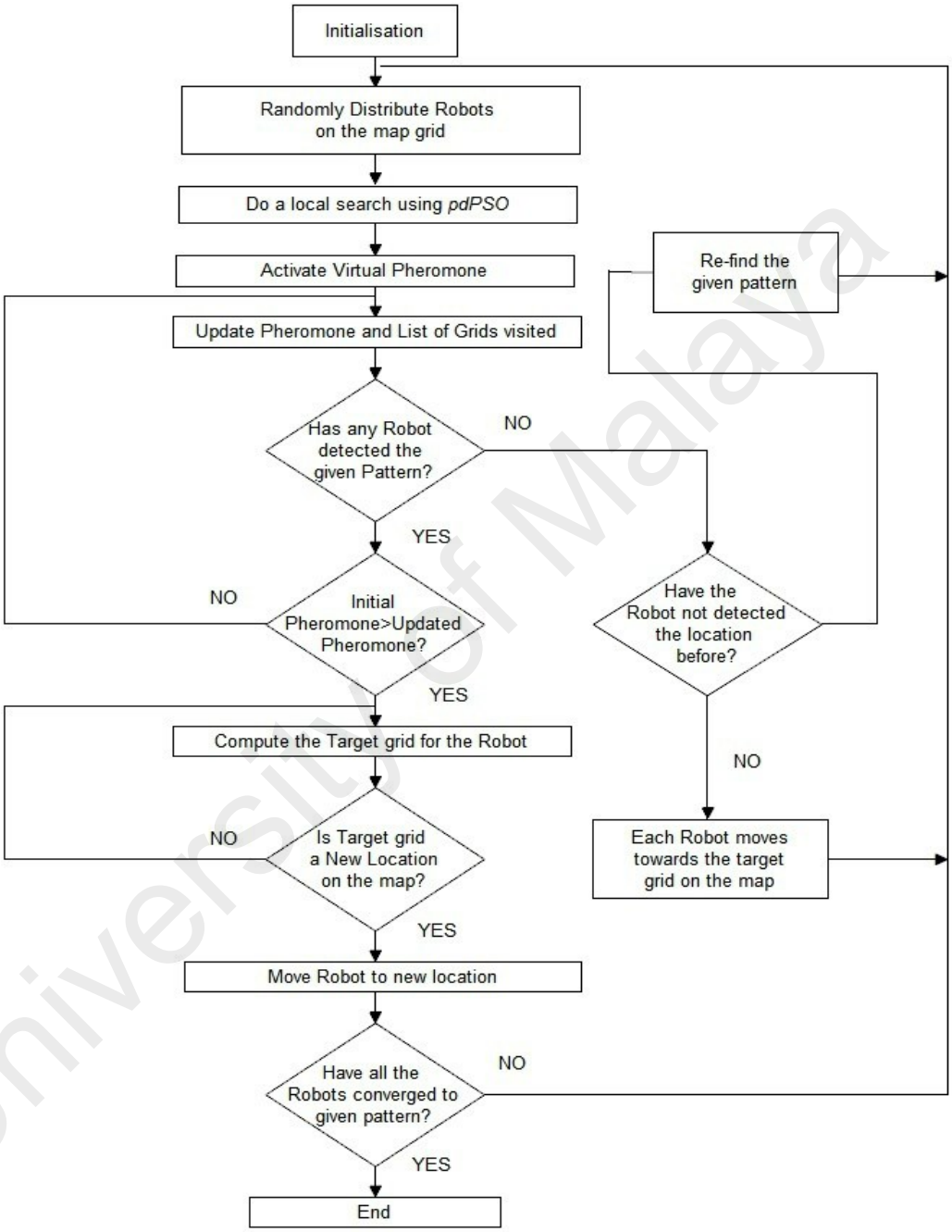


Figure 5.29: Flowchart of Pattern Formation Algorithm

5.3.4 Pattern Formation Results

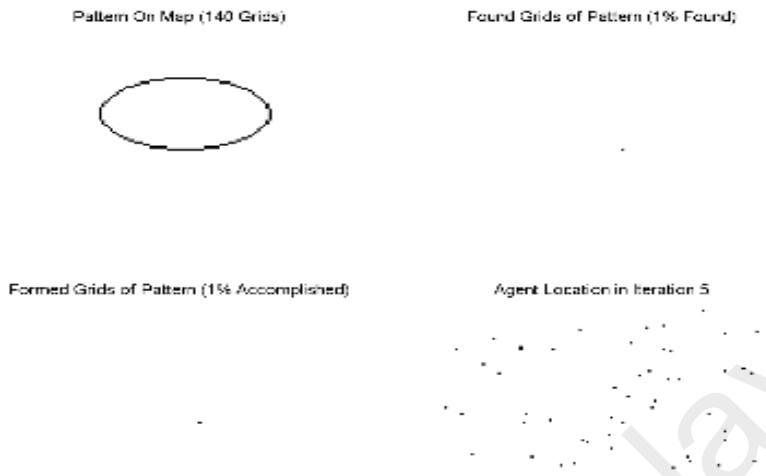


Figure 5.30 (a) 5th iteration

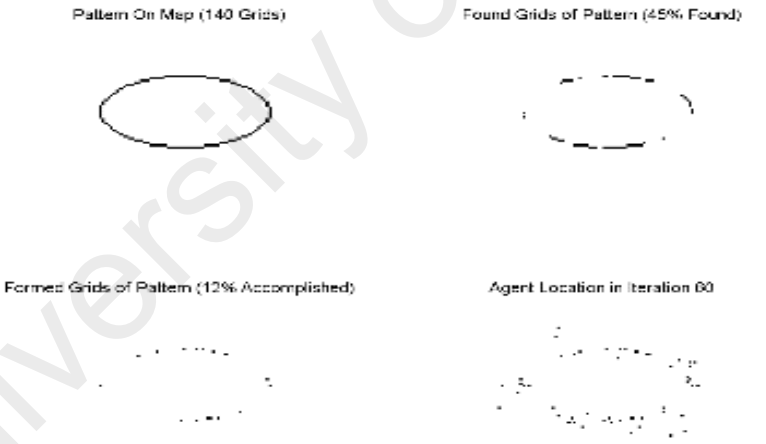


Figure 5.30 (b) 60th iteration

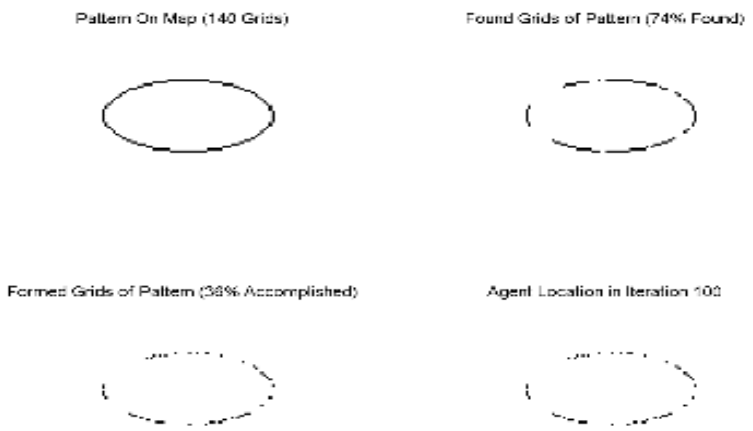


Figure 5.30 (c) 100th iteration

Figure 5.30 (a) – (c): 50-agent pattern formation, black spots stands for the robots

Figure 5.31 (a) 5th iteration

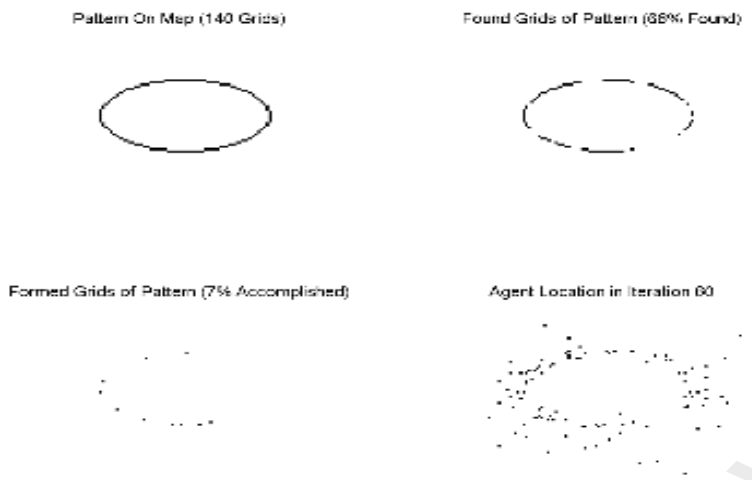


Figure 5.31 (b) 60th iteration

Figure 5.31 (c) 100th iteration

Figure 5.31(a) – (c): 100-agent pattern formation, black spots stands for the robots

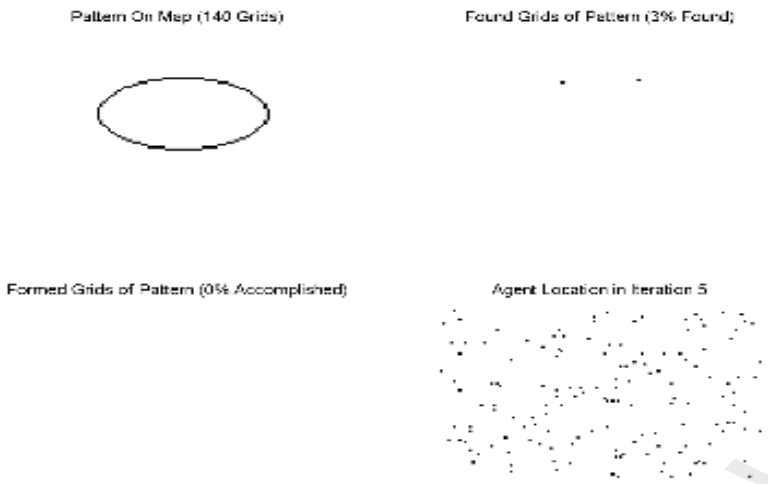


Figure 5.32 (a) 5th iteration

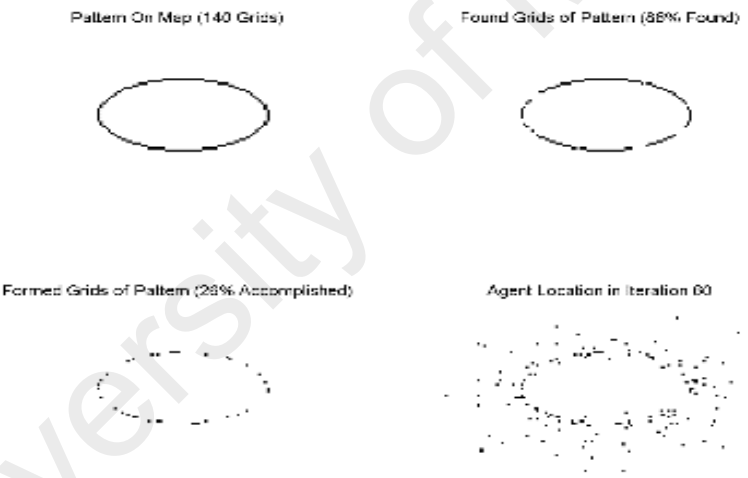


Figure 5.32 (b) 60th iteration

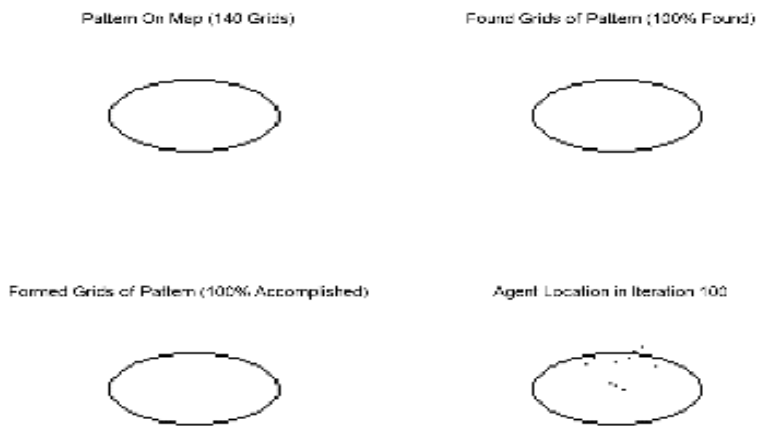


Figure 5.32 (c) 100th iteration

Figure 5.32 (a) – (c): 150-agent pattern formation, black spots stands for the robots

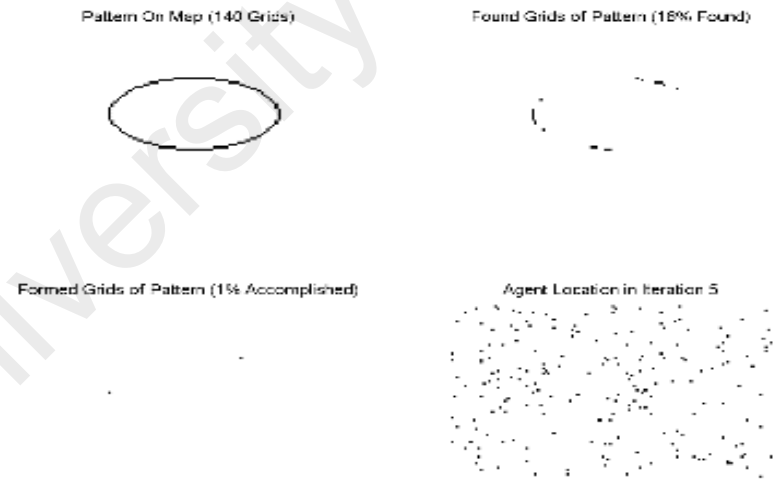


Figure 5.33 (a) 5th iteration

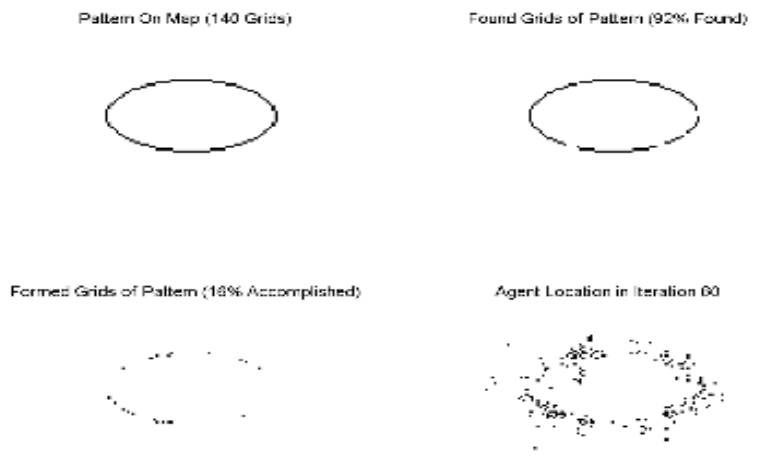


Figure 5.33 (b) 60th iteration

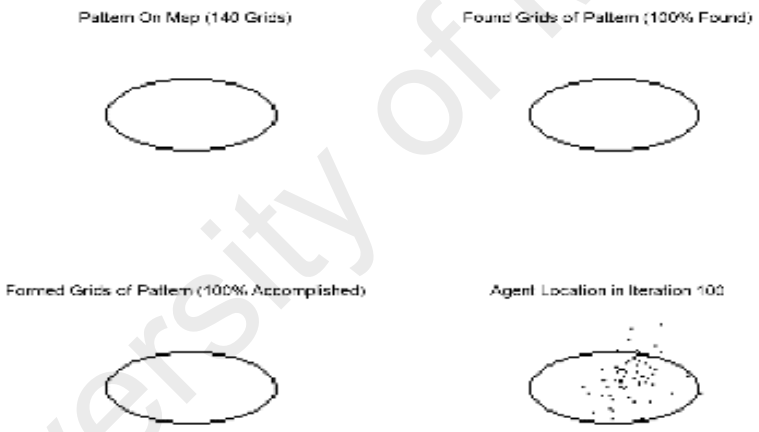


Figure 5.33 (c) 100th iteration

Figure 5.33 (a) – (c): 200-agent pattern formation, black spots stands for the robots

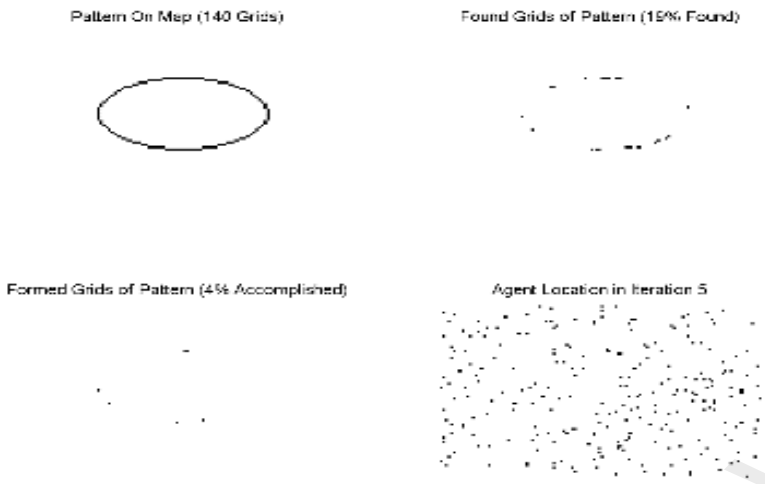


Figure 5.34 (a) 5th iteration

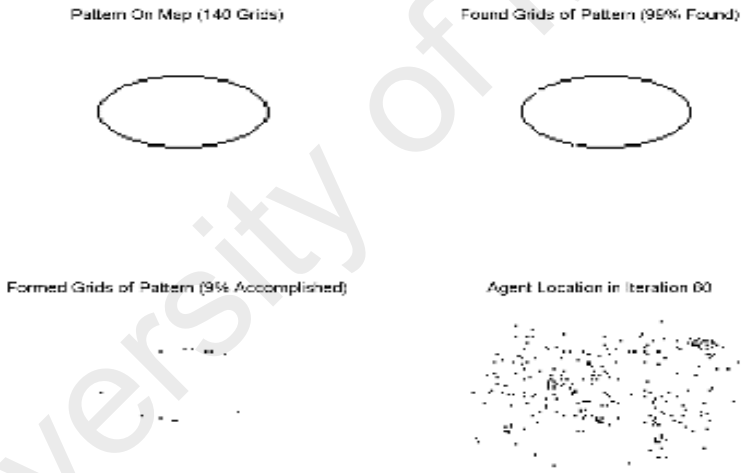


Figure 5.34 (b) 60th iteration

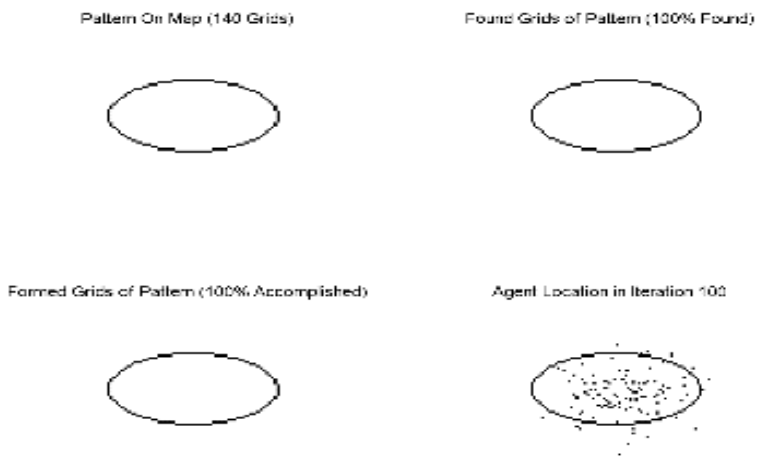


Figure 5.34 (c) 100th iteration

Figure 5.34 (a) – (c): 250-agent pattern formation, black spots stands for the robots

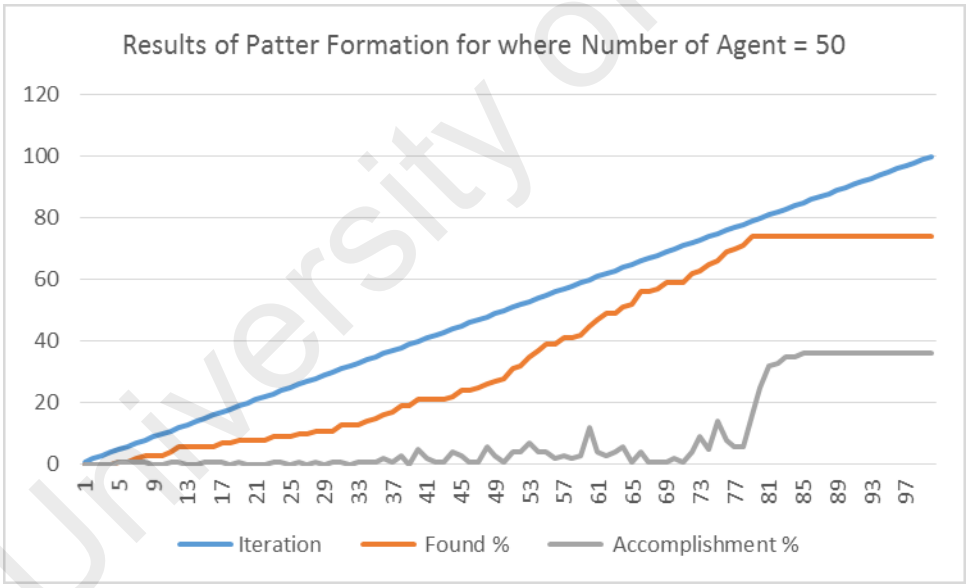


Figure 5.35: Graph of Pattern Formation using 50 agents

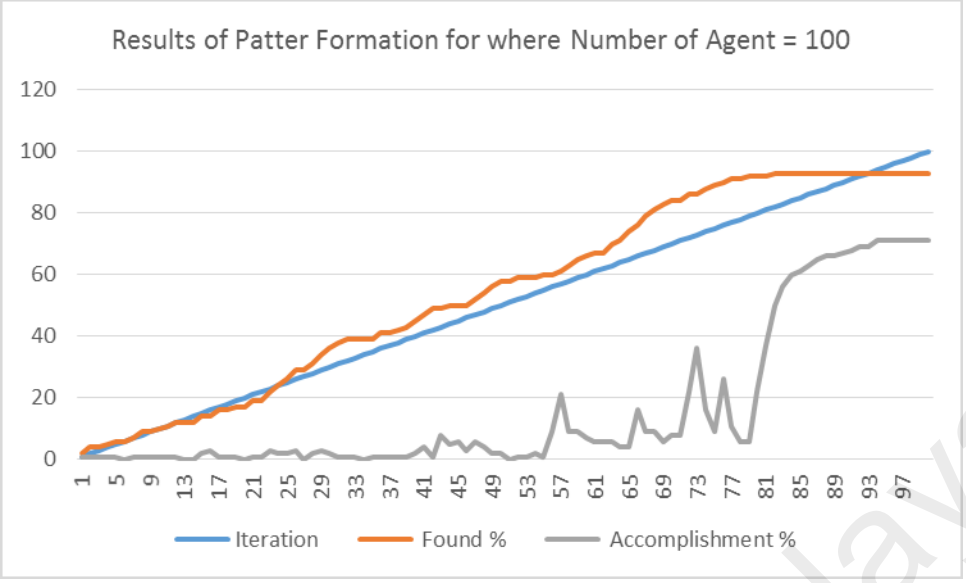


Figure 5.36: Graph of Pattern Formation using 100 agents

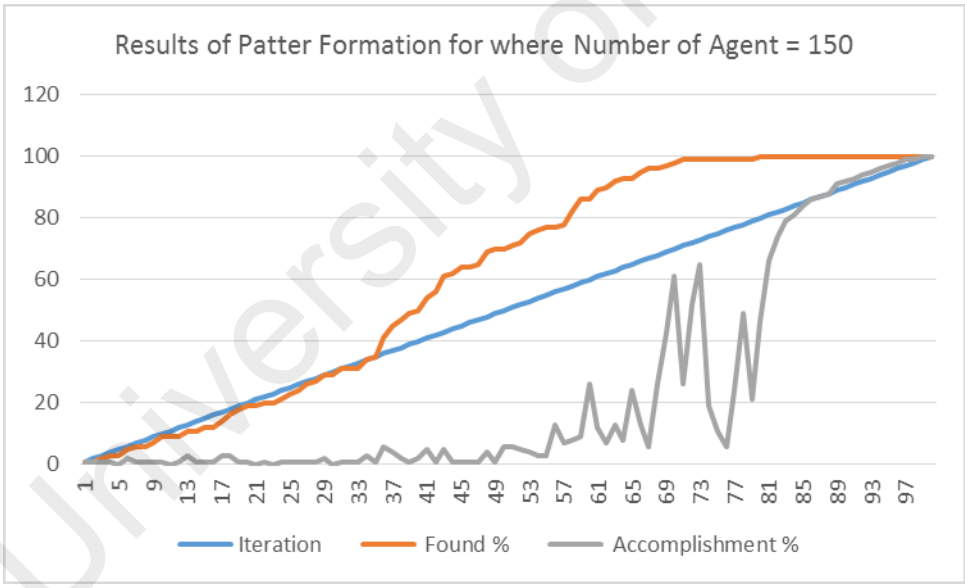


Figure 5.37: Graph of Pattern Formation using 150 agents

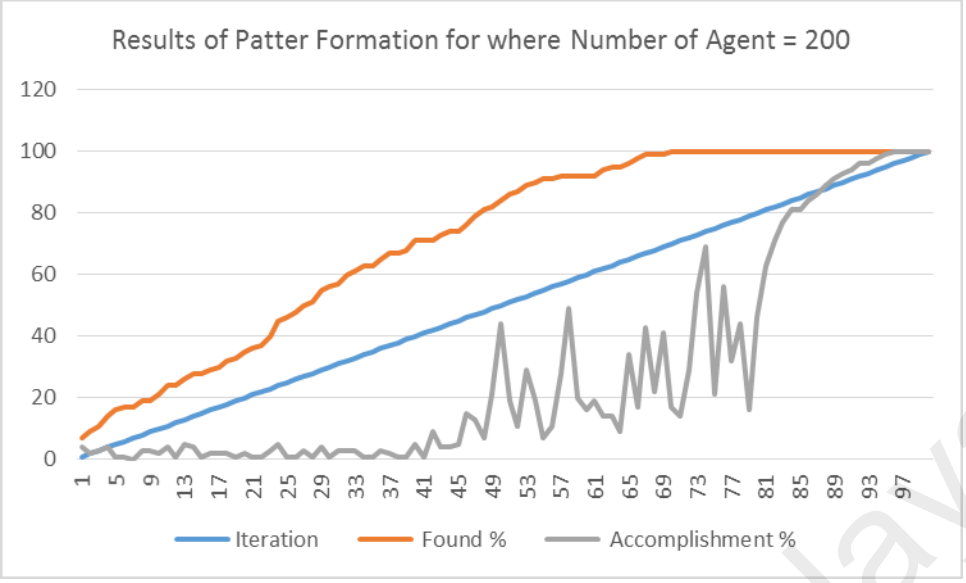


Figure 5.38: Graph of Pattern Formation using 200 agents

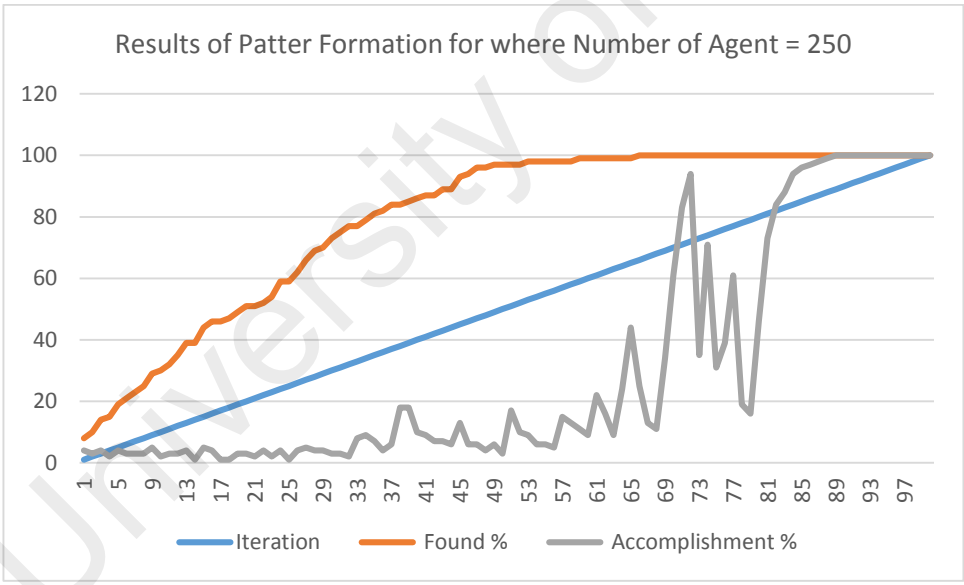


Figure 5.39: Graph of Pattern Formation using 250 agents

The results of our simulation of 50-agents, 100-agents, 150-agents, 200-agents, and 250-agents are presented in figures 5.30 – 5.34 and the tables in Appendix Z, Appendix AA, Appendix AB, Appendix AC, and Appendix AD. The figures illustrate the percentage (%) of

accomplishment in the process of forming Ellipse with different number of robots that comprise the swarm. We set the pheromone attenuation rate and pheromone accumulation rate to 0.01. The maximum value for pheromone is set to 1000. The PSO accelerate acceleration factor for local search and global search is set to 2 .0 and the inertia parameter to 0.1. As illustrated in the figures and tables above the percentage of accomplishment will get to 100% as the process continues except for where we have 50-agents and 100-agents. This shows that the numbers of agents have a great influence on the efficiency of our algorithm in forming the desired pattern. Our simulation results also signifies that the scalability capacity of our approach is commendable.

5.4 Summary of chapter

In this chapter the *pdPSO* and *pdAPSO* hybrid algorithm strategies were applied to solve flocking and pattern formation problems of swarm robots. These algorithms combine the explorative ability of PSO with the exploitative capacity of the Primal Dual Interior Point Method thereby possessing a strong capacity of avoiding premature convergence and making the robots to converge to a point and flock in real time. From the result of our simulations, the mean of the iteration for the agents to converge at the centre, and also flock from one zone to the other is almost the same. We decided to measure the number of iteration and not the time because the former is platform independent. This is an indication that the performance of our algorithm is good in terms of precision, convergence rate, equilibrium, robustness and ability to flock using homogenous set of swarm robots. Comparison was made among the flocking capacity of Primal Dual, PSO, APSO and *pdAPSO*. It was affirmed that *pdAPSO* performs better than all the other algorithms mentioned earlier.

We also applied one of our proposed algorithm to solve pattern formation problem in swarm robotics. For the sake of effective coordination and communication among the robots, the virtual pheromone was introduced. The use of *pdPSO* which is a PSO based technique ensures the efficiency of the pattern formation. We carried out some simulation was used to test the ability of our proposed algorithm to solve this problems and to evaluate its performance. We however have to deactivate the Primal-dual segment of our Primal-dual PSO algorithm to ensure increased efficiency and performance in the formation of the patterns. In our future work, we seek to extend the use *pdPSO* to solving flocking problems of swarm robotics with obstacle avoidance. Also, we intend implement our algorithms on hardware of physical swarm robots.

CHAPTER 6: CONCLUSION

6.1 Research Summary

Many variants of PSO suffers from stagnation and premature convergence (Gang et al. 2016) and are therefore unsuitable for solving problems that are related to Swarm robotics. This demonstrated that there is a fundamental defect in many variant PSOs and therefore necessitate the need to develop some more generalized PSO algorithm to support the implementation of dynamic tasks in swarm robotics. This is fundamental to controlling the development of customized PSO algorithms that can be used to address the primary defects of the existing PSO algorithm. Though swarm robotic are simple by their design, the task of coordinating a swarm of robots to accomplish a specific job (such as congregating, pattern formation, obstacle avoidance, flocking, segregating, exploring, mapping, and dispersion) can be very challenging. These types of jobs require a more dedicated algorithm unlike the conventional PSO algorithms that have been previously implemented in the field of robotics. To design a new algorithm that will meet with the above requirements, we did a survey of generic algorithm implementation for swarm robotics applications. This is to enhance our understanding of the limitations of particle swarm algorithm for dynamic optimisation tasks. Experiments were carried out firstly to establish the ground truth of some existing PSO algorithms, and to determine if they are functioning as described in the literatures. The result of our experiments confirmed the presence of premature convergence, failure of PSO algorithms to handle dynamic environment effectively, and inability of particles to escape from local minima. Secondly, our experiments were designed to determine the global optimum and local optima of each of the three PSO (original PSO, APSO, and SPSO) variants under the different benchmarking functions. And to confirm the problem of the particles in PSO been trapped in the local optimal. Lastly, we validated the existence of premature

convergence problem in PSO algorithms. In general, we wanted to investigate the performances of the three PSO variants (using the global optimum and local optimal as our yardstick of measuring performance) on the standard benchmark functions. Moreover we wanted to examine their convergence properties through these benchmark functions. The results of our simulations showed that PSO particles still got trapped in the local minima. And there is the presence of premature convergence in the variants of PSO that we experimented with. The results of our simulations are presented in figure 3.1 in chapter 3. It is evident from our simulation results, that in order to apply PSO into any swarm robotics implementation, we have to develop (or customized) the different flavour of PSO algorithms and thus contributed towards the ever-expanding pool of PSO algorithms. This should have not been the norm since the natural characteristics of the algorithm should be able to support any swarm robotics project. It is obvious that the algorithm is unsuitable for solving some optimisation problems.

6.1.1 Summary of Primal Dual Particle Swarm Optimisation (*pdPSO*)

We developed a new hybrid algorithm optimisation algorithm called Primal Dual Interior Point Method Particle Swarm Optimisation (*pdPSO*). This algorithm fuses the searching capability of PSO with the manipulative ability of the Primal Dual Interior Point Method thus having a robust capacity of evading premature convergence. A comparative study of the algorithm was done with the conventional PSO and Primal Dual method using nine benchmark functions. From the results of our simulations in chapter 4, it shows that our algorithm performs better in terms of precision, rate of convergence, steadiness and robustness. The behaviour of *pdPSO* under the unimodal and multimodal functions shows that the algorithm will be a suitable tool in solving complicated optimisation problems that PSO alone or Primal Dual alone cannot solve efficiently. In terms of speed of convergence,

pdPSO was faster than PSO and Primal-dual algorithms in 5 benchmark functions out of the of 9 functions used for the experiment. The *pdPSO* can therefore be termed as a fast algorithm that can proffer solution to complex optimisation tasks. The *pdPSO* also have a higher level of stability compared to the PSO and APSO algorithms. We hereby summarize that *pdPSO* is a very stable, and is capable of generating good results that are dependable. Lastly we observed that *pdPSO* is more robust than PSO and Primal-dual since it was able to successfully locate the global optimum on each of the benchmarking functions we used for our experiment specifically on some functions which are known to be very problematic for many of the state of the art optimisation algorithms. We conclude by saying that *pdPSO* is an algorithm that can survive unfavourable conditions.

The *pdPSO* algorithm was applied to solve the flocking and pattern formation problems of swarm robots. From the result of our simulations, the mean of the iteration for the agents to converge at the centre, and also flock from one zone to the other is almost the same. We decided to measure the number of iteration and not the time because the former is platform independent. The statistical results of our simulation as presented in Appendix C – F shows that there is no the mean of the iteration for the robots to converge at the centre and then to flock from one zone to other zones is almost the same. This clearly demonstrate that *pdPSO* have good convergence rate, accuracy, and the capacity to flock using identical group of robots.

We also applied *pdPSO* algorithm to solve pattern formation problem in swarm robotics. To ensure successful organization and interaction among the robots, the virtual pheromone was introduced. The use of *pdPSO* which is a PSO based technique ensures the efficiency of the pattern formation. The result of our Pattern formation simulation is presented in Appendixes Z – AD. Figures 5.34 – 5.38 show how the patterns are formed with

the number of iterations done. The graph showing the summary of the statistical results of the patterns formed and the number of iteration based on the number of robots involved in the pattern formation task is in Figures 5.39 – 5.43 of chapter 5. We hereby conclude that *pdPSO* is scalable as there is increase in the success rate the pattern formed with increase in the number of robots.

The performance of *pdPSO* was compared with eleven state of the art PSO algorithms listed in table 4.17 using twelve benchmark functions. The result of our experiments showed that *pdPSO* possesses the ability to efficiently overcome premature convergence problem and avoid being trapped in local minima on majority of the test functions. The efficiency of *pdPSO* becomes much more visible with the rotation and the shift benchmark functions. The comparisons shows that the integration of Primal-Dual into PSO is beneficial to enhancing the performance of PSO. We hereby conclude that apart from *pdAPSO* which produced the most accurate result, *pdPSO* have a superior performance compared to the other PSO variants on majority of the rotated functions and on the two shifted functions.

6.1.2 Summary of Primal Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*)

The second algorithm that we presented in this thesis is Primal Dual Asynchronous Particle Swarm Optimisation (*pdAPSO*). This algorithm integrates the exploring capability of PSO with the exploiting ability of the Primal Dual Interior Point Method. It thereby combined the strength of the two algorithms and so possessing a better capacity avoid particles been trapped in the local minima, and also the avoidance of premature convergence.

We did a comparison of our new algorithm (*pdAPSO*) with the PSO, APSO and *pdPSO* using seven (7) benchmark functions. From the experimental results in chapter 4, the performance of *pdAPSO* is better in terms of accuracy, convergence speed, reliability and

robustness when compared to some variants of PSO. Just like what we observed in *pdPSO,* the behaviour of *pdAPSO* under the unimodal and multimodal functions shows that the algorithm will also be another appropriate tool in solving complicated optimisation problems that PSO alone or Primal Dual alone cannot solve efficiently. It should be noted that there is no significant statistical distinction between the performance of *pdAPSO* and *pdPSO.*

The *pdAPSO* was also applied to solve the problem of flocking in swarm robotics. Our simulation results revealed that there is great similarity between the mean of convergence of the iteration at the centre, and the number of iteration to flock from one zone to another zone. The statistical results of our simulation are presented in Appendix G – J. It confirms that there is no difference in the mean of the iteration for the robots to converge at the centre and then to flock from one zone to other zones. This plainly illustrations that *pdAPSO* also have a high convergence speed, robustness, accuracy, and ability to flock using set of robots that are homogenous.

We compared the performance of *pdAPSO* with the other 11 state of the art PSO algorithms using twelve benchmark functions. The outcome of our experiments revealed that *pdAPSO* have a mean dependability of 80.4%. This is a sign that the integration of Primal Dual method into PSO algorithm will increase the reliability of PSO in solving the premature convergence problem and thereby converging to global optima. The comparisons also shows that the hybridisation of Primal-Dual and APSO helps in producing a more robust and dependable PSO algorithm. We conclude by saying that *pdAPSO* produced the most accurate result compared to the other PSO variants on majority of the rotated functions and on the two shifted functions. Moreover, the speed of convergence of *pdAPSO* algorithms is superior to that of other PSO algorithms on all the benchmark functions. Our algorithm called *pdAPSO* uses the smallest number of FEs to achieve acceptable solutions in all the 12 benchmark functions. This also proves that the Primal Dual method has augmented the PSO algorithm

in creating improved fitness value and generating higher diversity in the swarm population to increase the convergence speed of PSO particles.

6.2 Conclusion

The PSO algorithm, is a promising algorithm for providing solutions to different optimisation problems (Gang et al., 2016). Having discussed the suitability of PSO as an optimisation algorithm to solve swarm robotics problems in chapter two of this thesis, it uncovered some serious drawbacks that characterised the modern trends of developing and adapting a new bio-inspired algorithm for different swarm robotic tasks. These drawbacks can be adequately tackled by devising some other methods to increase the performance of PSO thereby enabling it to adequately handle dynamic optimisation tasks. We proposed the relevance of fusing Interior Point Method optimisation (Luke, 2010) and PSO to solve certain problems that are related to the existing variants of PSO which have discussed in this thesis. Such problem include premature convergence, the challenge of some of the particles been trapped in the local minima, and unsuitability of PSO for dynamic tasks. From the experiments conducted, it is obvious that the fusion of Primal Dual method and PSO helps to increase the reliability to PSO in preventing premature convergence of particles and thereby ensuring converging to global optima. Also, Primal Dual method has enhanced the performance of PSO algorithm in generating better fitness value and producing diversity in the swarm population to enhance the convergence speed of PSO particles.

The Interior-Point Method is a very popular optimisation algorithm that is widely known for its ability to solve large-scale linear problems effectively (Laird, 2006). The *pdPSO* and *pdAPSO* algorithms have been develop (in chapter 4) to solve the above-mentioned problems that are associated with PSO. The hybrid of Primal Dual and PSO provided a better balance between exploration and exploitation. This have solved the problem

of particles experiencing premature convergence and inability to escape being trapped in local minima thereby yielding superior results. The *pdPSO* and *pdAPSO* can be described as algorithms that have the ability to solve difficult optimisation problems quickly as demonstrated in chapter 6. They also have a higher level of stability compared to other variants of PSO algorithms. They are stable, robust, and capable of producing good and reliable results. Their robustness have helped them to be able to survive under benchmarking functions that many variants of PSO cannot survive when tested on them. We have been able to develop novel algorithms (*pdPSO* and *pdAPSO*) for swarm robotics cooperative movement. The results of our experiments in chapters 4 and 5 demonstrate that *pdPSO* and *pdAPSO* have high convergence rate, good accuracy, are robust, and they are suitable for control of cooperative movements in swarm robotics.

6.3 Future Directions

The concept of the hybridization of particle swarm optimisation (PSO) and Interior Point Method (IPM) as cooperative movement control algorithm in swarm robotics has been presented in this thesis. Many research in the field of swarm robotics have been centered on applications in the area of aggregation (Soysal & Sahin, 2005), box-pushing (Şahin, 2005), collective mapping (Jadbadaie, Lin, & Morse, 2003), flocking (Meng, Kazeem & Muller, 2007), foraging (Bayindir & Şahin, 2007) and (Campo et. al., 2010), pattern formation (Huaxing, et al., 2010), and segregation (Reynolds, 1987), soccer tournaments (Yang, et al, 2008), site preparation (Kim, Wang & Shin, 2006), and sorting (Jeschke, Liu & Schilberg, 2011).

In direction to the future, we intend to improve on the performance of *pdPSO* and *pdAPSO*. This is because it is not on all the benchmark functions that the performance of the algorithms were very high since there is no algorithm that performs very well on all known

benchmark functions. There is no algorithm that is suitable for solving all know problems. It thereby follows that our algorithms were basically designed to solve some of the drawbacks of the current variants of PSO. Moreover, our algorithms were successfully used to solve the swarm robotics tasks of aggregation, flocking, and pattern formation. However, for pattern formation we have to deactivate the Primal-dual segment of our Primal-dual PSO algorithm to ensure increased efficiency and performance in the formation of the patterns. In our future work, we seek to extend the use *pdPSO* to solving flocking problems of swarm robotics with obstacle avoidance and also implement our algorithms on hardware of physical swarm robots. Our future research would be to apply our *pdPSO* and *pdAPSO* to flocking, pattern formation, and foraging in swarm robotics while having obstacle avoidance in mind.

One of the principal weaknesses of the research work in this thesis is that the proposed model and approaches are yet to be executed and the performance verified on physical robots. There are times in which a solution may work perfectly during simulation but might not produce efficient result when tried on physical robots. Our intension is to implement our proposed algorithms on physical robots in the near future. Nevertheless, the proposed algorithms in this thesis should perform effectively on real world swarm robot (let's assume the robots can transfer information to nearby robots if they are close) since our viewpoint is that of cooperative movement. This means that all the robots will be affected by any disturbance from the environment. In the future, we intend to implement the proposed solution in real world swarm robotic systems.

REFERENCES

Abdel-Kader, R.F. (2010). Genetically improved PSO algorithm for efficient data clustering, in: *Proceedings of International Conference on Machine Learning and Computing,* pp. 71–75.

Abraham, A., Konar, A., & Das, S. (2008). Particle Swarm Optimisation and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives. *Studies in Computational Intelligence, (SCI)* 116, 1–38.

Abraham, A., Pant, M., Bouvry, P., & Thangaraj, R. (2011). Particle swarm optimisation: Hybridization perspectives and experimental illustrations, *Appl. Math. Comput,* vol. 217, 5208 – 5226, doi:10.1016/j.amc.2010.12.053

Aguirre, A. H., Muñoz Z. A. E., Diharce E. V. & Botello, R. S. (2007). COPSO: Constraints Optimisation via PSO algorithm, *Comunicación Técnic, (CC/CIMAT)*.

Akat, S. B. & Gazi, V. (2008) Particle swarm optimisation with dynamic neighborhood topology: three neighborhood strategies and preliminary results, in: *IEEE swarm intelligence symposium,* St Louis, USA, pp 1–8.

Al-Hassan, W., Fayek, M. B. & Shaheen, S. I. (2007). PSOSA: An optimized particle swarm technique for solving the urban planning problem, In: *Computer Engineering and Systems, the 2006 International Conference on, IEEE*. pp 401–405.

Amé, J., Halloy, J., Rivault, C., Detrain, C., & Deneubourg, J. L. (2006). Collegial decision making based on social amplification leads to optimal group formation. *Proceedings of the National Academy of Sciences,* 103(15):5835–5840.

Andries, P. Engelbrecht (2005). *Fundamentals of Computational Swarm Intelligence*, Wiley 1st edition. ISBN-10: 0470091916, 672 pages.

Akbari R & Ziarati K. (2011). A rank based particle swarm optimization algorithm with dynamic adaptation. *J Comput Appl Math,* vol. 235, pp. 2694–2714

Arkin, R. C. (1998). *Behaviour-Based Robotics*, MIT Press.

Armand, P., Gilbert, J. C. & Jan-Jégou, S. (2000). A feasible BFGS interior point algorithm for solving strongly convex minimization problems. *SIAM J. Optimisation*, vol. 11, pp. 199–222.

Ashish Raj (1994). *Evolutionary Optimisation Algorithms for Nonlinear Systems,* Thesis Submitted to the Department Electrical and Computer Engineering, Utah State University, Logan, Utah. 1994

Ashish, R. (1994). *Evolutionary Optimisation Algorithms for Nonlinear Systems,* Thesis Submitted to the Department Electrical and Computer Engineering, Utah State University, Logan, Utah.

Auger A & Hansen N (2005). Performance evaluation of an advanced local search evolutionary algorithm. In: *Proceedings of the IEEE congress on evolutionary computation,* vol. 1772, pp 1777–1784.

Aziz, N. A. A. & Ibrahim, Z. (2012). Asynchronous Particle Swarm Optimisation for Swarm Robotics, *International Symposium on Robotics and Intelligent Sensors (IRIS 2012). Procedia Engineering,* vol. 41 pp. 951 – 957.

Bai, Q. (2010). Analysis of Particle Swarm Optimisation Algorithm, *Computer and Information Science,* vol. 3 no 1, February, pp. 180 – 184.

Balch T. & Arkin, R. C. (1999). Behavior-based Formation Control for Multi-robot Teams, *IEEE Trans. on Robotics and Automation*.

Basterrechea, J., & Perez, J. R. (2005). Particle swarm optimisation and its application to antenna far field-pattern prediction from planar scanning. *Microwave and optical technology letters*, vol. 44(5), pp. 398–400.

Basturk, B., & Karaboga, D. (2007). A powerful and Efficient Algorithm for Numerical Function Optimisation: Artificial Bee Colony (ABC) algorithm, *J. Glob Optim,* vol. 39, 459-471.

Bayindir, L. & Şahin, E. (2007). A Review of Studies in Swarm Robotics. *Turk. J. Elec. Engin*. 15(2):115–147.

Beasley D. B., David R. Bull. & Ralph, R. Martin. (1993). An overview of Genetic Algorithms, *University Computing*, Vol. 15, No. 2 and 4, pp.58-69 and 170-181.

Belta C. & Kumar V. (2002). Trajectory design for formations of robots by kinetic energy shaping. In *Proceedings. ICRA '02. IEEE International Conference on Robotics and Automation*, vol.3, pp. 2593 – 2598.

Ben Kr¨ose & Patrick van der Smagt (1996). *An introduction to neural networks.* The University of Amsterdam.

Beni, G. (2004). From Swarm Intelligence to Swarm Robotics, in: *Proceedings of International Conference on Swarm Robotics*, pp. 1-9.

Ben-Tal, A. & Nemirovski, A. (2001). Lectures on Modern Convex Optimisation: Analysis, Algorithms, and Engineering Applications. *SIAM*.

Besdok, E., & Civicioglu, P., (2013). A conceptual comparison of the Cuckoo-search, particle swarm optimisation, differential evolution and artificial bee colony algorithms *Artif Intell Rev,* 39, (pp. 315-346). DOI 10.1007/s10462-011-9276-0.

Bhuvaneswari, R., Sakthivel, V.P., Subramanian, S., & Bellarmine, G. T. (2009). Hybrid approach using GA and PSO for alternator design, in: *Proceedings of the IEEE International Conference* Southeastcon, Atlanda, 2009, pp. 169–174.

Binitha S, & Sathya Siva S (2012). A Survey of Bio inspired Optimisation Algorithms. *International Journal of Soft Computing and Engineering* (IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, pp 137-151.

Boettcher, S. & Percus, A.G. (1999). Extremal optimisation: methods derived from coevolution, in: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 825–832.

Boggs, P.T. & Tolle, J.W. (1995). Sequential quadratic programming, *Acta Numerica*, vol. 4, Cambridge University Press, Cambridge, 1995. pp. 1–52.

Bonabeau, E., Sobkowski, A., Theraulaz, G., & Deneubourg, J.L. (1997). Adaptive task allocation inspired by a model of division of labor in social insects. *In Narayana, A., editor, Biocomputing and emergent computation: Proceedings of BCEC97*, pp 36–45, London, UK. World Scientific Press.

Boyd, S. & Vandenberghe, L. (2004), *Convex Optimisation*, Cambridge University Press, New York, 1st edition.

Brand, M., Masuda, M., Wehner, N. & Yu, X. H. (2010). Ant Colony Optimisation Algorithm for Robot Path Planning, *International Conference on Computer Design and Applications (ICCDA), IEEE,* Volume 3, page 436 – 440.

Brian B. (2014). Retrieved from http://www.mathworks.com/matlabcentral/file_exchange/7506-particle-swarm-optimisation-toolbox/content/testfunctions/

Brutschy, A., Pini, G., Pinciroli, C., Birattari, M., & Dorigo, M. (2012). Self-organized task allocation to sequentially interdependent tasks in swarm robotics. *Autonomous Agents and Multi-Agent Systems*, 2012:1–25.

Campo, A. (2011). *On the Design of Self-Organized Decision Making in Robot Swarms*. PhD thesis, IRIDIA, Université Libre de Bruxelles, Belgium.

Caponio, Neri, F. & Tirronen, V. (2009). Superfit control adaption in memetic differential evolution frameworks, *Soft Computing* 13, pp. 811–831.

Carlisle, A., & Dozier, G. (2001). An off-the-shelf PSO. In: *Workshop on Particle Swarm Optimisation*.

Chen, F., Chen, Z., Liu, Z., Xiang, L., and Yuan, Z. (2008). Decentralized formation control of mobile agents: A unified framework, *Physical A: Statistical Mechanics and its Applications*, Vol. 387, Issues 19–20, August 2008, Pages 4917-4926, ISSN 0378-4371, 10.1016/j.physa.2008.04.018.

Chen, M. R., Li, X., Zhang, X. & Lu, Y Z. (2010). A novel particle swarm optimizer hybridized with extremal optimisation, *Applied Soft Computing*, 10, pp. 367– 373.

Chena, M. R., Lia, X., Zhanga, X. & Lu, Y. Z. (2010). A novel particle swarm optimizer hybridized with extremal optimisation. *Applied Soft Computing,* vol. 10, Issue 2, pp. 367–373.

Chouzenoux, E., Idier, J., & Moussaoui, S. (2011). Efficiency of Line Search Strategies in Interior Point Methods for Linearly Constrained Signal Restoration. In*: Proceedings of the IEEE Workshop on Statistical Signal Processing (SSP*), Nice, France, pp.101-104.

Cieliebak, M. & Flocchini, P. (2002). Gathering autonomous mobile robots, In: *Proceeding 9th Int. Colloq. on Struct. Info. and Commun. Complex*, page 57-72.

Clerc, M. & Kennedy, J. (2002). The particle swarm optimisation: explosion, stability, and adaptive particle swarm optimisation. *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58–73.

Clerc, M. (2004). Semi-Continuous Challenge. Retrieved from clerc.maurice.free.fr/pso /semi-continuous_challenge/semi-continuous_challenge.htm

Conradie, E., Miikkulainen, R., Aldrich, C. (2002). Intelligent process control utilising symbiotic memetic neuro-evolution, *Proceedings of the IEEE Congress on Evolutionary Computation,* pp. 623 – 628.

Couceiro, M. S., Rocha, R. P. & Ferreira, N. M. F. (2011). Ensuring Ad Hoc connectivity in distributed search with Robotic Darwinian swarms, in: *Proceedings of the IEEE*

International Symposium on Safety, Security, and Rescue Robotics, SSRR2011, Kyoto, Japan, pp. 284–289.

D. E. Goldberg & J. H. Holland (1988), "Genetic algorithms and machine learning," *Machine learning,* vol. 3, no. 2, pp. 95–99.

Dada, E. G., & Ramlan, E. I. (2015). Primal-Dual Interior Point Method Particle Swarm Optimisation (*pdipmPSO*) Algorithm. In: *3rd Int'l Conference on Advances in Engineering Sciences & Applied Mathematics (ICAESAM'2015)*, London (UK), pp. 117-124.

Das, S., Biswas, A., Dasgupta, S., & Abraham, A. (2009). Bacterial Foraging Optimisation Algorithm: Theoretical Foundations, Analysis, and Applications*,* Volume 203/2009 *of Studies in Computational Intelligence*. Springer Berlin/Heidelberg, pp. 23-55.

David, E. G. (1994). Genetic and Evolutionary Algorithms come of age, Commun. *ACM*, Vol. 37, No. 3, pp.113-119, March 1994.

Davis, L. (1991). The Handbook of Genetic Algorithms. *Van Nostrand Reinhold*, New York.

Davoud, S. & Ellips M. (2009). Particle Swarm Optimisation Methods, Taxonomy and Applications. *International Journal of Computer Theory and Engineering*, Vol. 1, No. 5, December 2009.

de Berg M, Cheong O, van Kreveld M, & Overmars M (2008). *Computational Geometry Algorithms and Applications*. XII, 386 p. 370 illus., Hardcover ISBN: 978-3-540-77973-5

Del, V. Y, Venayagomoorthy, G. K., Mohagheghi, S., Hernandez, J. C. & Harley, R. G. (2008). Particle swarm optimisation: basic concepts, variants and applications in power systems. *IEEE Trans Evol Comput* 12(2):171 – 195.

Derr, K. & Manic, M. (2009). Multi-robot, multi-target particle swarm optimisation search in noisy wireless environments. In: *Proceedings of the 2nd conference on human system interactions,* Catania, Italy, pp 78–83.

Dervis, K. & Bahriye, B. (2007). A powerful and Efficient Algorithm for Numerical Function Optimisation: Artificial Bee Colony (ABC) algorithm. *J. Glob Optim* 39:459-471.

Dikin, I. (1967). Iterative solution of problems of linear and quadratic programming (in Russian), Doklady Akademia Nauk *SSSR*, vol. 8, pp. 674–675.

Doctor, S., Venayagamoorthy, G. & Gudise, V. (2004). Optimal PSO for collective robotic search applications. In: *IEEE congress on Evolutionary Computation,* pp 1390–1395.

Dorigo, M., Maniezzo, V. & Colorni, A. (1996). Ant System: Optimisation by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Vol. 26, No. 1, pp 29–41.

Ducatelle, F., Di Caro, G. A., Pinciroli, C., Mondada, F., & Gambardella, L. M. (2011). Communication assisted navigation in robotic swarms: self-organization and cooperation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, IEEE Computer Society Press, Los Alamitos, CA, pages 4981–4988.

Duniasn, P. (1996). *Autonomous Robots Using Artificial Potential Fields*, PhD Thesis. Universiteit Eindhoven. Page 1793-8201.

Dušan Glavaški, Mario Volf, & Mirjana Bonković (2009). Robot motion planning using exact cell decomposition and potential field methods. Proceedings of *the 9th WSEAS International Conference on simulation, modelling and optimisation.*

Easton, K. & Burdick, J. (2005). A coverage algorithm for multi-robot boundary inspection, in *Proceeding of the IEEE International Conference on Robotics and Automation,* ICRA, Barcelona, Spain, page 727 – 734.

Eberhart, R. C. & Shi, Y. (2000). Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimisation, In: *Proceedings of the Congress on Evolutionary Computation* Vol. 1 pp. 84 – 88.

Eberhart, R. C. & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, pp. 94-97.

Eberhart, R. C. & Shi, Y. (2002). Tracking and optimizing dynamic systems with particle swarms, In Evolutionary Computation, 2001. *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 94–100.

Eberhart, R. C., & Shi, Y. (1998). Parameter Selection in particle swarm optimisation, in: *Proceedings of the 7th International Conference on Evolutionary Programming,* Washington DC., pp. 591 – 600.

Egerstedt M. & Hu X. (2001). Formation constrained multi-agent control. *IEEE Transactions on Robotics and Automation*, 17(6):947–951.

Elor, Y., & Bruckstein, A. M. (2011). Uniform multi-agent deployment on a ring, *Theoretical Computer Science*, Vol. 412, Issues 8–10, 4 March 2011, Pages 783-795, ISSN 0304-3975, 10.1016/j.tcs.2010.11.023.

Emilie Chouzenoux, Saïd Moussaoui, & Jérôme Idier (2011). Efficiency of Line Search Strategies In Interior Point Methods For Linearly Constrained Signal Restoration. 2011 IEEE Statistical Signal Processing Workshop (SSP).

Esmin, A.A., Lambert-Torres, G., & Alvarenga, G.B. (2006). Hybrid evolutionary algorithm based on PSO and GA mutation, in: *Proceedings of 6th International Conference on Hybrid Intelligent Systems*, pp. 57–62.

Fan, S.K. & Zahara, E. (2007). A hybrid simplex search and particle swarm optimisation for unconstrained optimisation, European Journal of Operational Research 181, pp. 527–548.

Fan, S.K., Liang, Y. C. & Zahara, E. (2004). Hybrid simplex search and particle swarm optimisation for the global optimisation of multimodal functions, *Engineering Optimisation* 36, pp. 401–418.

Ferrante Eliseo (2013). Information Transfer in a Flocking Robot Swarm. *UNIVERSITÉ LIBRE DE BRUXELLES, Ecole Polytechnique de Bruxelles*, IRIDIA - Institut de Recherches Interdisciplinaires, et de Développements en Intelligence Artificielle, page 17-18.

Fiacco A. V. & Mccormick G. P. (1990). *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, N. Y., 1968. Reprinted by *SIAM Publications.*

Fiacco, A. V. & McCormick, G. P. (1967). *The sequential unconstrained minimization technique (SUMT) without parameters,* vol. 15, no. 5, pp. 820–827, September.

Fonseca, C. D. M, & Fleming, P. J. (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimisation. *Evolutionary Computation,* vol 3:1, pp 1-16.

Fourie, P.C. & Groenwold, A. A. (2002). The particle swarm optimisation algorithm in size and shape optimisation. *Structural and Multidisciplinary Optimisation*, 23(4):252–267.

Frisch, K. R. (1995). The logarithmic potential method of convex programming, *Technical Report, University Institute of Economics*, Oslo, Norway.

Gandelli, F. Grimaccia, Mussetta, M., Pirinoli, P., & Zich, R.E. (2007). Development and validation of different hybridization strategies between GA and PSO, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2782–2787.

Gandelli, Grimaccia F., Mussetta, M., Pirinoli, P., & Zich, R.E. (2005). Genetical swarm optimisation: a new hybrid evolutionary algorithm for electromagnetic application, in: *Proceedings of the 18th International Conference on Applied Electromagnetics*, ICECcom, Dubrovnik, Croatia, 2005, pp. 269–272.

Gandelli, Grimaccia F., Mussetta, M., Pirinoli, P., & Zich, R.E., (2006). Genetical swarm optimisation: an evolutionary algorithm for antenna design, *Journal of AUTOMATIKA* 47 (3-4), pp. 105–112.

Gang Xu, Binbin Liu, Jun Song, Shuijing Xiao & Aijun Wu (2016). Multiobjective sorting-based learning particle swarm optimisation for continuous optimisation. *Nat Comput*, DOI 10.1007/s11047-016-9548-3

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. Swarm Intelligence, 1(1):3–31.

Gasparri, A., & Prosperi, M. (2008). A Bacterial Colony Growth Framework for Collaborative Multi-Robot Localization, *IEEE International Conference on Robotics and Automation* Pasadena, CA, USA, May 19-23, 2008, pp. 2806 – 2811.

Gautam, A. & Mohan, S. (2013). A distributed algorithm for circle formation by multiple mobile robots, in *Proceedings of International Conference on Control, Automation, Robotics and Embedded Systems*, pp. 1-5, 16-18.

Ge, H. W., Sun, L., Liang, Y. C. & Qian, F. (2008). An effective PSO and AIS-based hybrid intelligent algorithm for job-shop scheduling, *IEEE Transactions on Systems, Man, And Cybernetics - Part A: Systems and Humans* 38, pp. 358–368.

Gengqian Liu, Tiejun Li, Yuqing Peng & Xiangdan Hou (2005). The Ant Algorithm for Solving Robot Path Planning Problem, *Information Technology and Applications,* third International Conference on Information Technology and Applications (ICITA'05), vol. 2, no. , pp. 25, 27.

George, A., Koh, B. I., Fregly, B. & Haftka, R. (2006). Parallel asynchronous particle swarm optimisation. *International Journal for Numerical Methods in Engineering*, vol. 67, pp. 578–595.

George, M. C. J., Carmelo, J. A. B. & Fernando, B. D. (2012). Volitive Clan PSO - An Approach for Dynamic Optimisation Combining Particle Swarm Optimisation and Fish School Search, Theory and New Applications of Swarm Intelligence, Dr. Rafael Parpinelli (Ed.), ISBN: 978-953-51-0364-6, *InTech* , Available from:http://www.intechopen.com/books/theory-and-new-applications-of-swarmintelligence/volitive-clan-pso-an-approach-for-dynamic-optimisation-combining-particle-swarm-optimisationand-fis

Gilbert, J. C., Armand, P., Jan-Jégou, S. (2000). A feasible BFGS interior point algorithm for solving strongly convex minimization problems, *SIAM J. Optimisation*, vol. 11, pp. 199 – 222.

Goldberg D E, Deb K, & Clark J H (1992). Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems* 6, pp 333-362.

Goldberg, D. E., Deb, K. & Clark, J. H. (1993). Accounting for noise in the sizing of populations. In *L.D Whitley, editor, Foundations of Genetic Algorithms* 2, Morgan Kaufmann, San Francisco, pp 127–140.

Grana, M., Duro, R. d'Anjou, A., & Wang, P. P. (2004). Information Processing with Evolutionary Algorithms: from *Industrial Applications to Academic Speculations, Springer* – ISBN 18523338660, pp 319.

Grimaccia, Mussetta, M., Pirinoli, P. & Zich, R.E. (2007). Genetical swarm optimisation: self-adaptive hybrid evolutionary algorithm for electromagnetic, *IEEE Transactions on Antennas and Propagation* 55 (3), pp. 781–785.

Grimaldi, E.A., Grimacia, F., Mussetta, M., Pirinoli, P., & Zich, R.E. (2004). A new hybrid genetical – swarm algorithm for electromagnetic optimisation, in: *Proceedings of International Conference on Computational Electromagnetics and its Applications*, Beijing, China, pp. 157–160.

Grosan, Abraham, A. & Nicoara, M. (2005). Search optimisation using hybrid particle sub-swarms and evolutionary algorithms, *International Journal of Simulation Systems, Science and Technology* 6, pp. 60–79.

Guerra, F. A., & Coelho, L. D. S. (2007). Applying Particle Swarm Optimisation to Adaptive Controller. A. Saad et al. (Eds.): *Soft Computing in Industrial Applications*, ASC, vol. 39, pp. 82–91.

Guo, Q. J., Yu, H. B. & Xu, A.D. (2006). A hybrid PSO-GD based intelligent method for machine diagnosis, *Digital Signal Processing* 16, pp. 402–418.

Hao, Z. F., Gua, G. H. & Huang, H. (2007). A particle swarm optimisation algorithm with differential evolution, in: *Proceedings of Sixth International Conference on Machine Learning and Cybernetics*, pp. 1031–1035.

Hayes, A. T., Martinoli, A., & Goodman, R. M. (2003). Swarm robotic odor localization: off-line Optimisation and Validation with Real Robots, *Robotica*, 21(4):427–441.

He, Q. & Wang, L. (2007). A hybrid particle swarm optimisation with a feasibility-based rule for constrained optimisation, Applied Mathematics and Computation 186, pp. 1407–1422.

Hendtlass, T. (2001). A Combined Swarm differential evolution algorithm for optimisation problems, in: *Proceedings of 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Lecture Notes in Computer Science*, vol. 2070, Springer Verlag, pp. 11–18.

Hendtlass, T., & Randall, M. (2001). A survey of ant colony and particle swarm meta-heuristics and their application to discrete optimisation problems, in: *Proceedings of the Inaugural Workshop on Artificial Life*, pp. 15–25.

Hereford, J. M. & Siebold, M. (2008) Multi-Robot Search using a Physically Embedded Particle Swarm Optimisation, *Int. Journal of Comput. Intell. Res*. 4(2):197–209.

Hereford, J. M. & Siebold, M. A. (2010). Bio-Inspired Search Strategies for Robot Swarms, Swarm Robotics from Biology to Robotics, Ester Martinez Martin (Ed.), ISBN: 978-953-307-075-9, *InTech*. Available from: http://www.intechopen.com/books/swarm-robotics-from-biology-to-robotics/bio-inspired-searchstrategies-for-robot-swarms.

Hereford, J. M. (2006). A distributed particle swarm optimisation algorithm for swarm robotic applications, In *IEEE congress on Evolutionary Computation*, pp 1678–1685.

Hoare, D. J., Krause, J., Peuhkuri, N. & Godin, J. G. J. (2000). Body size and shoaling in fish. *Journal of Fish Biology*, 57(6):1351–1366.

Holland, O. E. & Melhuish, C. (1999). Stigmergy, Self-Organization, and Sorting in Collective Robotics, *Artificial Life*, Vol. 5, pp. 173-202.

Holland, O. E. & Melhuish, C. (1999). Stigmergy, Self-Organization, and Sorting in Collective Robotics, *Artificial Life*, Vol. 5, pp. 173-202, 1999.

Howard, A., Matari c, M. J., & Sukhatme, G. S. (2002). Mobile sensor network deployment using potential fields: a distributed, scalable solution to the area coverage problem. In *Proceedings of the 2002 International Symposium on Distributed Autonomous Robotic Systems (DARS 2002)*, Piscataway, NJ. IEEE Press, pages 299–308.

Hu, X. & Eberhart, R. C. (2002). Adaptive particle swarm optimisation: detection and response to dynamic systems, *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, pp.l666-1670.

Hu, X. & Eberhart, R. C. (2002). Multi objective optimisation using dynamic neighborhood particle swarm optimisation. *Proc. of the IEEE/ CEC*, pp. 1677-1681.

Huang H, Qin H, Hao Z & Lim A (2012). Example-based learning particle swarm optimisation for continuous optimisation. *Inf Sci,* vol. 182 pp. 125–138.

Ikemoto Y. & Hasegawa, Y., Fukuda, T., Matsuda, K. Gradual spatial pattern formation of homogeneous robot group‖, *Information Sciences*, Vol. 171, Issue 4, 13 May 2005, Pages 431-445, ISSN 0020-0255, 10.1016/j.ins.2004.09.013.

J. C. Gilbert, P. Armand & S. Jan-Jégou (2000). A feasible BFGS interior point algorithm for solving strongly convex minimization problems, *SIAM J. Optimisation*, vol. 11, pp. 199 – 222.

J. Ren, K.A. McIsaac, R.V. Patel, & T. M. Peters, (2006). A Potential Field Model Using Generalized Sigmoid Functions, IEEE Trans. on Systems, Man, and Cybernetics, Part B: Cybernetics, vol.37, pp.477–484, 2006.

Jadbadaie, A., Lin, J. & Morse, A.S. (2003). Coordination of Groups of Mobile Autonomous Agents Using Nearest Neighbor Rules. *IEEE Transactions on Automatic Control* 48(6), 988–1001.

Jäger, M. & Nebel, B. (2002). Dynamic decentralized area partitioning for cooperative cleaning robots, in *Proceeding of IEEE International Conference on Robotics and Automation, ICRA,* Washington DC, USA, page 3577 – 3582.

Jatmiko, W., Sekiyama, K. & Fukuda, T. (2006). A Mobile Robots PSO-based for Odor Source Localization in Dynamic Advection-Diffusion Environment. *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and System*s, October 9 – 15, Beijing, China. Page 4527 - 4532.

Jatmiko, W., Sekiyama, K. & Fukuda, T. (2007). A PSO-based mobile robot for odor source localization in dynamic advection-diffusion with obstacles environment: theory, simulation and measurement. *IEEE Comput Intell Mag,* 2(2):37–51.

Jeong, S., Hasegawa, S., & Shimoyama, K., & Obayashi, S. (2009). Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real-world design optimisation, *IEEE Computational Intelligence Magazine*, pp. 36–44.

Jeschke, S., Liu, H. & Schilberg, D. (2011). Swarm Robot Flocking: An Empirical Study. (Eds.): *ICIRA 2011*, Part II, LNAI 7102, pp. 495–504.

Jian, M., & Chen, Y. (2006). Introducing recombination with dynamic linkage discovery to particle swarm optimisation, *Proceedings of the Genetic and Evolutionary Computation Conference,* 85–86.

John H. Holland (1992). Adaptation in natural and artificial systems, *MIT Press*, Cambridge, MA.

Johnson, C. A. Seidel, J. & Sofer, A. (2000). Interior-point methodology for 3-D PET reconstruction," *IEEE Trans. Medical Imaging*, vol. 19, no. 4, April 2000.

Jorge Nocedel & Stephen J. Wright (2006). Numerical Optimisation, 2nd Edition. *Springer Science + Business Media, LLC.*

Jose, G. N., Alba, E. & Apolloni, J. (2009). Particle swarm hybridized with differential evolution: black box optimisation benchmarking for noisy functions, in: *Proceedings of International Conference Genetic and Evolutionary Computation*, pp. 2343–2350.

Juan R. V., Zhang, M. & Winston, S. (2011). A Performance Study on Synchronous and Asynchronous Updates in Particle Swarm Optimisation, *GECCO'11*, July 12–16, Dublin, Ireland. Pp.: 21 – 28.

Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimisation for recurrent network design, *IEEE Transactions On Systems, Man, And Cybernetics-Part B: Cybernetics* 34, pp. 997–1006.

Jun-jie, Xu. and Zhan-hong, Xin., (2005) "An extended particle swarm optimizer", *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05),* 1530-2075/05.

Kao, Y. T., Zahara, E. & Kao, I. W. (2008). A hybridized approach to data clustering, *Expert Systems with Applications,* 34, pp. 1754–1762.

Kao, Y.T., & Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimisation for multimodal functions, *Applied Soft Computing* 8, pp. 849–857.

Kavehand, S. T. (2009). A particle swarm ant colony optimisation for truss structures with discrete variables, *Constructional Steel Research*, 65, pp. 1558–1568.

Keisam Thoiba Meetei (2014). A Survey: Swarm Intelligence vs. Genetic Algorithm. *International Journal of Science and Research (IJSR),* ISSN (Online): 2319-7064, Impact Factor (2012): 3.358.

Kenned, J. & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm, *Int. IEEE Conf. on Systems, Man, and Cyber*, Vol.5, pp.4104 – 4108.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimisation, in: *Proceedings of the International Conference on Neural Networks*, vol. 4, IEEE Press, Piscataway, NJ, pp. 1942-1948.

Kennedy, J., Eberhart, R. & Shi, Y. (2001). *Swarm Intelligence*, Morgan Kaufmann Publishers, US.

Kennedy, J., Eberhart, R. & Shi, Y. (2004) *Swarm Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA.

Khamsawang, S., Wannakarn, P. & Jiriwibhakorn, S. (2010). Hybrid PSO-DE for solving the economic dispatch problem with generator constraints, in: *Proceedings of the IEEE International Conference on Computer and Automation Engineering*, vol. 5, pp. 135–139.

Khatib, O. (1968). Real time obstacle avoidance for manipulators and mobile robots, *International Journal of Rob. Res.* 5 (1): 90-98.

Kim, D. H., Wang, H. & Shin, S. (2006). Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Function-Analytical Design Guidelines. *J. Int. Robot Systems* 45, 36–394.

Kim, H., (2006). Improvement of genetic algorithm using PSO and Euclidean data distance, *International Journal of Information Technology* 12, pp. 142–148.

Koh, B. I., Fregly, B. J., George, A. D. & Haftka, R. T. (2005). Parallel asynchronous particles swarm for global biomechanical, *Int Journal of Number Methods Eng*., 67(4): 578–595.

Koo T.J. & Shahruz S.M. (2001). Formation of a group of unmanned aerial vehicles (UAVS). *Proceedings of the American Control Conference* Arlington, VA June 25-27, pp. 69-74.

Krieger, M. J. B., Billeter, J. B. & Keller, L. (2000). Ant-like Task Allocation and Recruitment in Cooperative Robots, *Nature*, Vol. 406, pp. 992-995.

Krink, T., & Løvbjerg, M. (2002). The lifecycle model: combining particle swarm optimisation, genetic algorithms and hill climbers, *Proceedings of the Parallel Problem Solving From Nature*, pp. 621–630.

Krishnanand K.N. & Ghose, D. (2005). Formations of minimaist mobile robots using local-templates and spatially distributed interactions, *Robotics and Autonomous Systems,* Vol. 53, Issues 3–4, 31. Pages 194-213, ISSN 0921-8890, 10.1016/j.robot.2005.09.006.

Kuo, I. H., Horng, S. J., Kao, T. W., Lin, T. L., Lee, C. L., Terano, T. & Pan, Y. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimisation model, *Expert Systems with Applications*, 36, pp. 7027–7032.

Laird, C. D., (2006). *Structured Large-Scale Nonlinear Optimisation Using Interior-Point Method: Applications in Water Distribution Systems.* PhD Thesis Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania.

Langer, J. S. (1980). Instabilities and pattern formation in crystal growth. Reviews of Modern Physics, vol 52, no 1, pp 1-28.

Lee, T. Y. (2007b) Optimal Spinning Reserve for a Wind-Thermal Power System Using EIPSO, *IEEE/ TPWRS*, Vol. 22, No.4, pp. 1612 – 1621.

Li, T., Xu, L. & Shi, X.W. (2008). A hybrid of genetic algorithm and particle swarm optimisation for antenna design, *PIERS* online 4, pp. 56–60.

Liang J. J, Suganthan P. N & Deb K (2005). Novel composition test functions for numerical global optimisation. In: *Proceedings of the IEEE swarm intelligence symposium*, pp 1–14.

Liang, J. J., & Qin, A. K., Suganthan, P. N. & Baskar, S. (2006). Comprehensive Learning Particle Swarm Optimizer for Global Optimisation of Multimodal Functions, *IEEE transactions on evolutionary computation,* Vol. 10, No. 3, June, pp. 281 – 295.

Liang, J. J., Qin, A. K., & Baskar, S. (2006). Comprehensive Learning Particle Swarm Optimizer for Global Optimisation of multimodal Functions, *IEEE Trans. Evolutionary Computation,* Vol. 10, No. 3.

Liu, G., Li, T., Peng, Y. & Hou, X. (2005). The Ant Algorithm for Solving Robot Path Planning Problem, *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05), IEEE*.

Liu, H. Cai, Z. & Wang, Y. (2009). Hybridizing particle swarm optimisation with differential evolution for constrained numerical and engineering optimisation, *Applied Soft Computing* 10, pp. 629–640.

Liu, H., Abraham, A. & Zhang, W. (2007). A fuzzy adaptive turbulent particle swarm optimisation, *International Journal of Innovative Computing and Applications* 1, pp. 39–47.

Liu, W., Winfield, A. F. T., Sa, J., Chen, J., & Dou, L. (2007). Towards energy optimisation: emergent task allocation in a swarm of foraging robots. Adaptive Behavior, 15(3):289–305.

Locatelli, M. (2003). A note on the Griewank test function, *Journal of Global Optimisation*, 25 (2), pp. 169-174, doi:10.1023/A:1021956306041

Luh, G. C. & Liu, W. W. (2004). Reactive Immune Network Based Mobile Robot Navigation, In G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, editors, *Proceeding of the Third Conference ICARIS*, *Springer*. pp. 119 – 132.

Luke Michael Blohm Winternitz (2010). *Primal-Dual Interior-Point Algorithms for Linear Programs with Many Inequality Constraints.* Dissertation submitted to the Faculty of the Graduate School of the University of Maryland, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Luo, J. & Zhang, Z. (2006). Research on the Parallel Simulation of Asynchronous Pattern of Particle Swarm Optimisation, *Computer Simulation*, 22(6):78–70.

Mahanti, G.K. & Chakrabarty, A. (2007). Phase-only and amplitude-phase synthesize of dual- pattern linear antenna arrays using floating-point genetic algorithms, Progress in Electromagnetics Research, PIER 68 pp. 247–259.

Mantas Paulinas & Andrius Ušinskas (2007). A survey of genetic algorithms applications for image enhancement and segmentation. *Information technology and control*, Vol.36, No.3, pp. 278 - 284.

Marco Dorigo & Thomas Stützle (2004). *Ant Colony Optimisation*. A Bradford Book, The MIT Press, Cambridge, Massachusetts, London, England.

Martinoli, A., Ijspeert, A. J. & Mondada, F. (1999). Understanding Collective Aggregation Mechanisms: From Probabilistic Modeling to Experiments with Real Robots, *Robotics and Autonomous Systems*, Vol. 29, pp. 51-63.

Mataric, M. J., Nilsson, M. & Simsarian, K. T. (1995). Cooperative Multirobot Box-Pushing, *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Matlab Central (2013). Retrieved from http://www.mathworks.com/matlabcentral/file exchange/_7506-particle-swarm-optimisation-toolbox/content/testfunctions/

Mehrotra, S. (1992). On the implementation of a primal-dual interior point method, *SIAM Journal on Optimisation*, vol. 2, pp. 575-601.

Mei Wang, Zhiyong Su, Dawei Tu, & Xichang Lu (2013). A Hybrid Algorithm Based on Artificial Potential Field and BUG for Path Planning of Mobile Robot. IEEE 2nd International Conference on Measurement, Information and Control, pp 1393 - 1398.

Meinhardt, H. (1982). Models of biological pattern formation, volume 6. Academic Press, London, UK.

Mendes, R., Kennedy, J. & Neves, J. (2004). The fully informed particle swarm: Simpler, may be better, *IEEE Transactions on Evolutionary Computation*, 8(3): page 204–210.

Miao, H. (2009). *Robot Path Planning in Dynamic Environments Using a Simulated Annealing Based-Approach*, MSc. Thesis Faculty of Science and Technology, Queensland University of Technology.

Minsky, M. (1967). Computation: Finite and Infinite Machines. *Prentice-Hall*, Upper Saddle.

Mo, W., Guan, S. U. & Puthusserypady, S. K. (2007). A novel hybrid algorithm for function optimisation: particle swarm assisted incremental evolution strategy, Studies in Computational Intelligence 75, pp. 101–125.

Mohammadi, A., & Jazaeri, M. (2007). A hybrid particle swarm optimisation-genetic algorithm for optimal location of SVC devices in power system planning, in: *Proceedings of 42nd International Universities Power Engineering Conference*, pp.1175–1181.

Montes de Oca, M. A., Ferrante, E., Scheidler, A., Pinciroli, C., Birattari, M., & Dorigo, M. (2011). Majority-rule opinion dynamics with differential latency: a mechanism for self-organized collective decision-making. Swarm Intelligence, 5(3–4):305–327.

Murthy, R., Arumugam, M.S. & Loo, C.K. (2009). Hybrid particle swarm optimisation algorithm with fine tuning operators, *International Journal of Bio-Inspired* Computation 1, pp. 14–31.

Navarro I. & Matía F. (2013). An Introduction to Swarm Robotics. *Hindawi Publishing Corporation ISRN Robotics*, Volume 2013, Article ID 608164, 10 pages.

Newton, I. (2010). The Migration Ecology of Birds. *Elsevier*, Amsterdam, Netherlands.

Nolfi, S. & Floreano, D. (2004). Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines. Intelligent Robots and Autonomous Agents. MIT Press, Cambridge, MA.

Omran, M., Engelbrecht, A.P., & Salman, A. (2008). Bare bones differential evolution, *European Journal of Operational Research* 196, pp. 128–139.

Panigrahi, B. K., Lim, M. H., & Shi, Y. (2011). Handbook of Swarm Intelligence: Concepts, Principles and Applications, *Springer-Verlag Berlin Heidelberg*, ISBN 978-3-642-17389-9, pp. 119-132.

Panigrahi, B. K., Shi, Y. & Lim, M. H. (2011). Handbook of Swarm Intelligence: Concepts, Principles and Applications, *Springer-Verlag* Berlin Heidelberg, ISBN 978-3-642-17389-9, pp.119-132.

Pant, M. Thangaraj, R. & Abraham, A. (2009). DE-PSO: a new hybrid meta-heuristic for solving global optimisation problems, *New Mathematics and Natural Computation*.

Pant, M., Thangaraj, R. & Abraham, A. (2008). Particle swarm optimisation using adaptive mutation, in: *Proceedings of 19th International Conference on Database and Expert Systems Application,* Italy, pp. 519–523.

Park, J. B., Shin, J. R., Lee, K. Y. & Jeong, Y. W. (2010). An Improved Particle Swarm Optimisation for Nonconvex Economic Dispatch Problems, *IEEE Transactions on Power Systems*, Vol. 25, No. 1, pp. 156- 166.

Parker C.A.C. & Zhang, H. (2006). Collective Robotic Site Preparation. *Adaptive Behavior.* Vol.14, No. 1, 2006, pp. 5-19.

Payton, D., Daily, M., Estowski, R., Howard, M. & Lee, C. (2001). Pheromone Robotics, Vol. 11, No. 3, November.

Peleg, D. (2005). Distributed Coordination Algorithms for Mobile Robot Swarms: A New Directions and Challenges, *IWDG 2005, LNCS 3741, Springer-Verlag Berlin Heidelberg*, page 1-12.

Perez, J. R. & Basterrechea, J. (2005). Particle swarm optimisation and its application to antenna far field-pattern prediction from planar scanning, *Microwave and optical technology letters,* 44(5):398–403.

Pinar, C. & Erkan B. (2013). A conceptual comparison of the Cuckoo-search, particle swarm optimisation, differential evolution and artificial bee colony algorithms. *Artif Intell Rev,* 39:315-346, DOI 10.1007/s10462-011-9276-0.

Poli, R., Langdon, W. B. & Holland, O. (2005b). Extending particle swarm optimisation via genetic programming. In M. Keijzer et al. (Eds.), Lecture notes in computer science: Vol. 3447. *Proc. of the 8th European conf. on genetic programming, Springer,* pp. 291–300.

Premalatha, K., & Natarajan, A.M. (2009). Discrete PSO with GA operators for document clustering, *International Journal of Recent Trends in Engineering* 1, pp. 20–24.

Pugh, J. & Martinoli, A. (2007). Inspiring and modeling multi-robot search with particle swarm optimisation, in *IEEE swarm intelligence symposium,* Honolulu, USA, pp 332–339.

Pugh, J., Segapelli, L. & Martinoli, A. (2006). Applying aspects of multi robot search to particle swarm optimisation, in *Proceedings of the 5th international workshop on ant colony optimisation and swarm intelligence*, Brussels, Belgium, pp. 506–507.

Quintana, V. H. & Torres, G. L. (1997). Introduction to Interior-Point Methods, *IEEE PES task Force on Interior-Point Methods Applications to Power Systems*, Available online at http://wathvdc3.uwaterloo.ca/~iee-ipm.

Rekleitis, I., Dudek, G. & Milios, E. (2001). Multi-robot collaboration for robust exploration, *Annals of Mathematics and Artificial Intelligence*, 31 (1-4) 7-14.

Renegar, J. (2001). A Mathematical View of Interior-Point Methods in Convex Optimisation. *SIAM.*

Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. In Stone, M. C., editor, SIGGRAPH '87: *Proceedings of the 14th annual conference on computer graphics and interactive techniques*, New York, NY. ACM Press, pages 25–34.

Richard H. B., Jorge N., & Richard A. W (2005). "Feasible Interior Methods Using Slacks for Nonlinear Optimisation." Retrieved from http://users.eecs.northwestern.edu/~rwaltz/articles/ feasible.pdf

Roberto B., Mauro B. & Srinivas P. (2005). *Do Not Be Afraid of Local Minima: Affine Shaker and Particle Swarm*. Department of Information and Communication Technology, Universita Degli Studi Di Trento. Retrieved from http://dit.unitn.it/

Ru, N., & Jianhua, Y. (2008). A GA and particle swarm optimisation based hybrid algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1047–1050.

Rusman, H. (2013). An Efficient Particle Swarm Optimisation Technique for Solving the Non-convex Economic Dispatch Problems. *International Journal of Engineering Sciences,* 2(5).

Ružica, M. G., Dragan, I. O., & Branko, M. K. (2007). Particle Swarm Optimisation Algorithm and Its Modifications Applied to EM Problems, *IEEE,* pp. 427- 430.

Şahin, E. Swarm Robotics (2005). From Sources of Inspiration to Domains of Application. In: Şahin, E., Spears, W. M. (eds.) Swarm Robotics (2004). *LNCS, Springer, Heidelberg*, vol. 3342, pp. 10–20.

Samuel, R., Nikolause, C. & Alcherio, M. (2009). Collaborative coverage using a swarm of networked miniature robots, *Journal of Robotics and Autonomous Systems,* vol. 57. Page 517 – 525.

Schutte, J. (2001). Particle Swarms in Sizing and Global Optimisation. Master's thesis, University of Pretoria, South Africa.

Sedlaczek K. & Eberhard P. (2005). Constrained Particle Swarm Optimisation of Mechanical Systems, *6th World Congresses of Structural and Multidisciplinary Optimisation*, Rio de Janeiro, 30 May - 03 June, Brazil.

Settles, M., & Soule, T. (2005). Breeding swarms: a GA/PSO hybrid, *Proceedings of Genetic and Evolutionary Computation Conference,* 161–168.

Sha, D.Y., & Hsu, C.Y. (2006). A hybrid particle swarm optimisation for job shop scheduling problem, *Computers & Industrial Engineering* 51, pp. 791–808.

Shelokar, P.S., Siarry, P., Jayaraman, V. K. & Kulkarni, B.D. (2007). Particle swarm and ant colony algorithms hybridized for improved continuous optimisation, *Applied Mathematics and Computation* 188, pp. 129–142.

Shen, Q., Shi, W. M. & Kong, W. (2007). Hybrid particle swarm optimisation and tabu search approach for selecting genes for tumor classification using gene expression data, *Computational Biology and Chemistry,* 32, pp. 53–60.

Shengli Ai & Yude Wang (2011). Application of Improved Genetic Algorithms in Structural Optimisation Design. *International Conference, ICCIC 2011, Wuhan, China, September 17-18, 2011*. Proceedings, Part VI pp 480-487. DOI 10.1007/978-3-642-24097-3_72

Shi, Y. & Eberhart, R. (2001). Particle Swarm Optimisation with Fuzzy Adaptive Inertia Weight, *Proc. of the Workshop on Particle Swarm Optimisation.* Indianapolis, IN.

Shi, Y. & Eberhart, R. C. (1998). Parameter Selection in particle swarm optimisation, In *Proceedings of the 7th International Conference on Evolutionary Programming,* pp. 591 – 600.

Shunmugalatha, A. & Slochanal, S.M.R. (2008). Optimum cost of generation for maximum loadability limit of power system using hybrid particle swarm optimisation, *International Journal of Electrical Power & Energy Systems*, 30, pp. 486–490.

Simin, M. O., Zeng, J. & Weibin, X. U. (2011). An Extended Particle Swarm Optimisation Algorithm Based On Self-Organization Topology Driven By Fitness. *Journal of Computational Information Systems,* pp. 4441-4454. Available at http://www.Jofcis.com

Simon Haykin (1999). *Neural Networks, A Comprehensive Foundation*. Prentice Hall, 2nd edition.

Sofer, A., Johnson, C. A., & Seidel, J. (2000). Interior-point methodology for 3-D PET reconstruction," *IEEE Trans. Medical Imaging*, vol. 19, no. 4, 271-285.

Song, S., Kong, L., Gan, Y. & Su, R. (2008). Hybrid particle swarm cooperative optimisation algorithm and its application to MBC in alumina production, *Progress in Natural Science*, 18 pp. 1423–1428.

Spector, L., Klein, J., Perry, C. & Feinstein, M. D. (2003). Emergence of collective behavior in evolving populations of flying agents. In E. Cantu-Paz et al. editor. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2003)*, Berlin, Germany, pp.61-73.

Stenz, A. (1994). Optimal and Efficient Path Planning for Partially-known Environments, *Proceedings of the IEEE International Conference on Robotics and Automation,* pages 3310 – 3317.

Stephanie Forrest & Melanie Mitchell (1993).What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation. *Machine Learning* 13, 285-319, 1993.

Suarez, J. & Murphy, R. (2011). A survey of animal foraging for directed, persistent search by rescue robotics, in: *Proceedings of the IEEE International Symposium on Safety, Security and Rescue Robotics*, Kyoto, Japan, pp. 314–320.

Suganthan P. N, Hansen N, Liang J. J, Deb K, Chen Y. P, Auger A & Tiwari S (2005). Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimisation. In*: Proceedings of the congress on evolutionary computation*, pp 1–50.

Sumpter, D. J. T. (2010). *Collective Animal Behavior*. Princeton University Press, Princeton, NJ.

Sun, J., Lai, C. H. & Jun, X. J. (2012). *Particle Swarm Optimisation: Classical and Quantum Perspectives*, CRC press, Taylor and Francis Group. ISBN: 978-1-4398-3576-0, pp 60-61.

Talbi, H., & Batouche, M. (2004). Hybrid particle swarm with differential evolution for multimodal image registration, *Proceedings of the IEEE International Conference on Industrial Technology,* vol. 3, pp. 1567–1573.

Talukder, S. (2011). *Mathematical modelling and applications of particle swarm optimisation*. Master's thesis, Blekinge Institute of Technology, The School of Engineering, 2011.

Tang, Q. & Eberhard, P. (2011). A PSO-based algorithm designed for a swarm of mobile robots, *Journal of Industrial Application. Struct Multidisc Optim.* vol. 44, page 483-498.

Ting, T.O., Wong, K.P. & Chung, C.Y. (2008). Hybrid constrained genetic algorithm/particle swarm optimisation load flow algorithm, in: *IET Proceedings of Generation, Transmission & Distribution*, vol. 2, pp. 800–812.

Torn, A. Z. (1989). Global Optimisation, *Lecture Notes in Computer Science*, vol. 350, Springer-Verlag.

Torres, G. L., & Quintana, V. H. (1997). Introduction to Interior-Point Methods, *IEEE PES task Force on Interior-Point Methods Applications to Power Systems*.

Trivedi, N., Lai, W. & Zhang, Z. (2001). Optimizing Windows Layout by Applying a Genetic Algorithm, *Proceedings of the 2001 Congress on Evolutionary Computation*, Seoul, Korea, pp. 431- 435.

Valdez, F., Melin, P. & Castillo, O. (2009). Evolutionary method combining particle swarm optimisation and genetic algorithms using fuzzy logic for decision making, in: *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 2114–2119.

Vandenberghe, L. & Boyd, S. (2004). *Convex Optimisation*. (1st edition). Cambridge University Press, New York.

Vanneschi, L. D., Codecasa, G. & Mauri (2011). A Comparative Study of Four Parallel and Distributed PSO Methods. *New Generation Computing*, *Ohmsha, Ltd. and Springe*r, vol 29, page 129 – 161.

Venayagomoorthy, G. K., Hernandez, J. C., Harley, R. G., Valle, Y. D., & Mohagheghi, S. (2008). Particle swarm optimisation: basic concepts, variants and applications in power systems. *IEEE Trans Evol. Comput,* 12(2), 171 – 195.

Vicsek, T., Czirok, A., Jacob, E. B., Cohen, I. & Schochet, O. (1995). Novel Type of Phase Transitions in a System of Self-Driven Particles. *Physical Review Letters*, 75, 1226–1229.

Victoire, T.A.A. & Jeyakumar, A.E. (2004). Hybrid PSO–SQP for economic dispatch with valve-point effect, *Electric Power Systems Research* 71, pp. 51–59.

Wang H, Sun H, Li C. H., Rahnamayan S, & Pan J. S. (2013) Diversity enhanced particle swarm optimization with neighborhood search. *Inf Sci*, vol. 223, pp. 119–135

Wang H F, Moon I, Yang S X & Wang D W (2012). A memetic particle swarm optimisation algorithm for multimodal optimisation problems. *Inf Sci,* vol. 197 pp. 38–52.

Ward, C. R., Gobet, F. & Kendall, G. (2001). Evolving collective behavior in an artificial ecology. *Artificial Life*, Vol. 7, No. 2, 2001, pp.191-209.

Wehenkel, L., & Glavic, V. (2004). *Interior Point Methods: A Survey, Short Survey of Applications to Power Systems, and Research Opportunities*. Technical Report. University of Liège Electrical Engineering and Computer Science Department Sart Titman B-28 4000 Liege, Belgium.

Wei, Z. He & Pei, W. (2002). Swarm directions embedded in fast evolutionary programming, in: *Proceedings of the World Congress on Computational Intelligence*, vol. 2, pp. 1278–1283.

Weigel, T., Gutmann, J. S., Dietl, M., Kleiner, A. & Nebel, B. (2002). CS Freiburg: Coordinating Robots for Successful Soccer Playing", Special Issue on Advances in Multi-Robot Systems, T. Arai, E. Pagello, and L. E. Parker, Editors, *IEEE Trans. on Robotics and Automation*, Vol. 18, No.5, pp. 685-699.

Winternitz, L. M. B. (2010). Primal-Dual Interior-Point Algorithms for Linear Programs with Many Inequality Constraints. *Dissertation submitted to the Faculty of the Graduate School of the University of Maryland*, College Park in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Wright, S. J. (1996). Primal-Dual Interior-Point Methods, *SIAM*, Philadelphia.

Xie, X. F., Zhang, W. J. & Yang, Z. L. (2002b). Adaptive Particle Swarm Optimisation on Individual Level, *Int. Conf. On Signal Processing (ICSP*), pp: 1215-1218.

Xu, G. (2013). An adaptive parameter tuning of particle swarm optimization algorithm. *Appl Math Comput,* vol. 219, pp.4560–4569

Xu H, Guan H, Liang A & Yan X. (2010). A Multi-robot Pattern Formation Algorithm Based on Distributed Swarm Intelligence. 2010 *Second International Conference on Computer Engineering and Applications.* doi:10.1109/iccea.2010.22

Xu, H., Guan, H., Liang, A., & Yan X. (2010). A Multi-Robot Pattern Formation Algorithm Based on Distributed Swarm Intelligence. *Second International Conference on Computer Engineering and Applications, IEEE.* DOI 10.1109/ICCEA.2010.22, pp. 77-75.

Xu, W. & Gu, X. (2009). A hybrid particle swarm optimisation approach with prior crossover differential evolution, in: *Proceedings of ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pp. 671–678.

Xue, S. D., Zhang, J. H. & Zeng, J. C. (2009). Parallel asynchronous control strategy for target search with swarm robots, *Int Journal of Bio-Inspired Comput (IJBIC*) 1(3):151–163.

Xue, S., Li, J., Zeng, J., He X. & Zhang, G. (2011). Synchronous and Asynchronous Communication Modes for Swarm Robotics Search, in Mobile Robots – Control Architectures, Bio-Interfacing, Navigation, Multi Robot Motion Planning and Operator Training, J. Bedkowski, Editor, *Intech*. Retrieved from http://www.intechopen.com/

Xue, S., Zan, Y., Zeng, J., Xue, Z., & Du, J. (2012). Group Decision Making Aided PSO-Type Swarm Robotic Search, *International Symposium on Computer, Consumer and Control, IEEE 2012.* Page 785 – 788.

Yamauchi, B. (1999). Decentralized Coordination for Multi-robot Exploration, *Robotics and Autonomous Systems*, Vol. 29, No. 1, pp. 111-118.

Yamauchi, B. (1999). Decentralized Coordination for Multi-robot Exploration, *Robotics and Autonomous Systems*, Vol. 29, No. 1, pp. 111-118.

Yang, Chen, Y., & Zhao, Z. (2007). A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimisation problems, in: *Proceedings of the IEEE International Conference on Control and Automation*, pp. 166–170.

Yang, X. S. (2012). Swarm-Based Metaheuristic Algorithms and No-Free-Lunch Theorems, Theory and New Applications of Swarm Intelligence, Dr. Rafael Parpinelli (Ed.), ISBN: 978-953-51-0364-6, *InTech* Available from: http://intechopen.com/books/theory-and-new-applications-of-swarm-intelligence/swarm-based-metaheuristic-algorithms-and-no-free-lunch-theorems

Yang, Y., Xiong, N., Chong, N.Y. & Défago, X. (2008). A Decentralized and Adaptive Flocking Algorithm for Autonomous Mobile Robots. In: *the 3rd International Conference on Grid and Pervasive Computing Workshops, IEEE Press*. pp. 262–268.

Yao, X., Bullinaria, J. A., Rowe, J., Tino, P. & Kaban, A. (2004). Parallel Problem Solving from Nature, PPSN VIII, *8th International Conference Birmingham, UK, September Proceedings*.

Yukiko Yamauchi (2013). A survey on pattern formation of autonomous mobile robots: asynchrony, obliviousness and visibility, *ELC International Meeting on Inference, Computation, and Spin Glasses (ICSG2013), Journal of Physics: Conference Series* 473 (2013) 012016, doi:10.1088/1742-6596/473/1/012016.

Zahara, Y. T. Kao (2009). Hybrid Nelder–Mead simplex search and particle swarm optimisation for constrained engineering design problems, *Expert Systems with Applications,* 36, pp. 3880–3886.

Zeng, J., Hu, J. & Jie, J. (2006). Adaptive Particle Swarm Optimisation Guided by Acceleration Information, *Proc. IEEE/ ICCIAS*, Vol.1, pp.351-355.

Zhan Z. H, Zhang J, Li Y & Shi Y. H (2011). Orthogonal learning particle swarm optimisation. *IEEE Trans Evol Comput* 15:832–847.

Zhang, C., Ning, J., Lu, S. Ouyang, D. & Ding, T. (2009). A novel hybrid differential evolution and particle swarm optimisation algorithm for unconstrained optimisation, *Operations Research Letters* 37 (2009) 117–122.

Zhang, N. & Wunsch, D. (2003a). Fuzzy logic in collective robotic search, in the *12th IEEE International Conference on Fuzzy Systems,* Vol. 2, page 1471 – 1475.

Zhang, N. & Wunsch, D. (2003b). A comparison of dual heuristic programming (DHP) and neural network based stochastic optimisation approach on collective robotic search problem, In *Proceedings of the International Joint Conference on Neural Networks*, vol. 1, page 248 – 253.

Zhang Yudong, Wu Lenan, & Wang Shuihua (2013). "UCAV path planning by fitness-scaling adaptive chaotic particle swarm optimisation," Mathematical Problems in Engineering, vol. 2013, page 1 – 9, Article ID 705238.

Zhao X. (2010). A perturbed particle swarm algorithm for numerical optimization. *Appl Soft Comput*., 10(1), pp. 119–124

Zheng, X., Jain, S., Koenig, S. & Kempe, D. (2005). Multi-robot forest coverage, in *Proceeding of the IEEE/RS. International Robots and Systems, IROS*, Edmonton, Alberta, Canada. Page 3852 – 3857.

APPENDIX A

LIST OF PUBLICATIONS AND PAPERS

1. Dada, E. G., & Ramlan, E. I. (2016). Understanding the limitations of particle swarm algorithm for dynamic optimisation tasks. ACM Computing Survey. *Status (Accepted for publication).*

2. Dada, E. G., & Ramlan, E. I. (2016). *pdPSO*: The Fusion of Primal-Dual interior point method and particle swarm optimisation algorithm. Malaysian Journal of Computer Science. *Status (Under Review).*

3. Dada, E. G., & Ramlan, E. I. (2016). Improving particle swarm optimization through better localization of dynamic search spaces: A primal-dual asynchronous PSO (APSO) as cooperative movement control algorithm. Applied Soft Computing. *Status* (*Under Review).*

Conference Papers

1. Dada, E. G., & Ramlan, E. I. (2015). Primal-Dual Interior Point Method Particle Swarm Optimisation (*pdipmPSO*) Algorithm. In: *3rd Int'l Conference on Advances in Engineering Sciences & Applied Mathematics (ICAESAM'2015)*, London (UK), pp. 117-124. (*Published*).

   Best oral presenter award

2. Dada, E. G., & Ramlan, E. I. (2015). A Hybrid Primal-Dual-PSO (pdipmPSO) Algorithm for Swarm Robotics Flocking Strategy. The Second International Conference on Computing Technology and Information Management (ICCTIM2015). Malaysia, IEEE. (*Published*).