Self-Limitation, Dynamic and Flexible Approaches for Particle Swarm Optimisation

Mohd Nadhir Ab Wahab

School of Computer, Science and Engineering

University of Salford

United Kingdom

Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

May 2017

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLE

ACKNOWLEDGEMENT

First of all, I would like to thank god almighty Allah S.W.T for the strengths and His blessing in completing this thesis. I would like to express my special appreciation and thanks to my advisor, Professor Samia Nefti-Meziani, you have been a tremendous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless. I would also like to thank my co-supervisor, Dr Edmund Chadwick for his advice and guidance which help me get through the challenge during my research.

I also want to thank you Dr Adham Atyabi for helping me in designing the experiments and given me valuable advice on research and life. Another person that I would like to thank is Mr Andy Baker, who help me with configuring the robot and always help me when I am in need. I also want to thank all members of Advanced Robotics and Autonomous Centre, University of Salford for helping me out and encouraging me to stay positive all the time.

Special thanks to my family. Words cannot express how grateful I am to all my family members' including my mother-in-law and father-in-law and a special dedication to my mother, Chek Noorlia Abu Bakar and my father, Ab Wahab Mat for all of the sacrifices that you've made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank all of my friends who supported me in writing and incented me to strive towards my goal. I would like express appreciation to my beloved wife Nor Hazwani who spent sleepless nights with and was always giving me her full support. Not to forget, my lovely daughter who cheering me up when I was down, Nur Kaisah.

LIST OF ABBREVIATIONS

ACO	ANT COLONY OPTIMISATION
AEPSO	AREA EXTENDED PARTICLE SWARM OPTIMISATION
CAEPSO	CONSTRICTED AREA EXTENDED PARTICLE SWARM OPTIMISATION
CMPSO	CONSTRICTED MORPHOLOGY PARTICLE SWARM OPTIMISATION
CPSO	CONSTRICTED PARTICLE SWARM OPTIMISATION
CSA	CUCKOO SEARCH ALGORITHM
DA	DIJKSTRA'S ALGORITHM
DAC	DYNAMIC ACCELERATION COEFFICIENTS
DE	DIFFERENTIAL EQUATION
DPPSO	DYNAMIC PARAMETERISING PARTICLE SWARM OPTIMISATION
EDP	ENGINEERING DESIGN OPTIMISATION PROBLEMS
FIW	FIXED INERTIA WEIGHT
GA	GENETIC ALGORITHM
LDIW	LINEAR DECREASING INERTIA WEIGHT
MPSO	MORPHOLOGY PARTICLE SWARM OPTIMISATION
PF	POTENTIAL FIELD
PRM	PROBABILISTIC ROAD MAP

PSO	PARTICLE SWARM OPTIMISATION
RANDIW	RANDOM INERTIA WEIGHT
RRT	RAPIDLY-EXPLORE RANDOM TREE
SI	SWARM INTELLIGENCE
TSP	TRAVELLING SALESMAN PROBLEM
TVAC	TIME-VARYING ACCELERATION COEFFICIENTS

LIST OF PUBLICATIONS

Ab Wahab, MN, Nefti-Meziani, S and Atyabi, A (2015), 'A comprehensive review of swarm optimization algorithms', PLoS ONE, 10 (5). (In Press) IF: 3.234

Ab Wahab, MN, Nefti-Meziani, S and Atyabi, A (2017), 'Comparison between Conventional and Evolutionary Algorithms in Mobile Robot Path Planning', PLoS ONE. (Under Review) IF: 3.234

Ab Wahab, MN, Nefti-Meziani, S, Chadwick, E and Atyabi, A (2017), 'Morphology Particle Swarm Optimization', Swarm Intelligence. (Ready to Submit) IF: 2.577

Ab Wahab, MN, Nefti-Meziani, S and Atyabi, A (2017), 'Dynamic based Behaviour Particle Swarm Optimization', Expert System with Applications. (Ready to Submit) IF: 2.980

ABSTRACT

Swarm Intelligence (SI) is one of the prominent techniques employed to solve optimisation problems. It has been applied to problems pertaining to engineering, schedule, planning, networking and design. However, this technique has two main limitations. First, the SI technique may not be suitable for the online applications, as it does not have the same aspects of limitations as an online platform. Second, setting the parameter for SI techniques to produce the most promising outcome is challenging. Therefore, this research has been conducted to overcome these two limitations. Based on the literature, Particle Swarm Optimisation (PSO) was selected as the main SI for this research, due to its proven performances, abilities and simplicity. Five new techniques were created based on the PSO technique in order to address the two limitations. The first two techniques focused on the first limitation, while the other three techniques focused on the latter. Three main experiments (benchmark problems, engineering problems, path planning problems) were designed to assess the capabilities and performances of these five new techniques. These new techniques were also compared against several other well-established SI techniques such as the Genetic Algorithm (GA), Differential Equation (DE) and Cuckoo Search Algorithm (CSA). Potential Field (PF), Probabilistic Road Map (PRM), Rapidly-explore Random Tree (RRT) and Dijkstra's Algorithm (DA) were also included in the path planning problem in order to compare these new techniques' performances against Classical methods of path planning. Results showed all five introduced techniques managed to outperform or at least perform as good as well-established techniques in all three experiments.

CHAPTER 1

INTRODUCTION

1.1 Background

Swarm Intelligence (SI) is one of the topics that have attracted an interest from many researchers in various fields. Bonabeau defined SI as '*the emergent collective intelligence of groups of simple agents*' (Bonabeau, Dorigo, & Theraulaz, 1999). SI is the cooperative intelligence manners of independent and decentralised systems, e.g., artificial clusters of simple agents. SI examples can be found in groups of social insects, social insects in nest-building, and joint arrangement and clustering. Two crucial theories that are considered as essential properties of SI are self-organisation and division of labour. Self-organisation is defined as the ability of a system to evolve its particles or agents into a suitable shape without any peripheral help. Bonabeau (Bonabeau et al., 1999) also stated that self-organisation depends on four essential properties; positive feedback, negative feedback, fluctuation and multiple interactions. These two types of feedback are useful for strengthening and balancing the swarm. Fluctuations are useful for randomising while multiple interactions arise when the swarms share knowledge among themselves within their searching space. The next property of SI is the partition of labour, which is defined as the simultaneous performance of various simple and doable tasks by agents. This partition permits the swarm to attack complex problems that require individuals to work together (Bonabeau, Dorigo, & Theraulaz, 1999; Kennedy, 2006; Y. F. Zhu & Tang, 2010).

1.2 Contributions of Work

Contributions of work towards body knowledge are important for each research conducted. It allows the knowledge to keep growing and not static over time. Therefore, these are the list of contribution gained from this research:

- Morphology Particle Swarm Optimisation (MPSO) and Constricted Morphology Particle Swarm Optimisation (CMPSO) have introduced a new option to weigh or control the influence of particle's best position and swarm's best position. This approach gives a self-limitation on every particle movement which as a result, makes the movement feasible for online application too.
- This research has also proven a little tweak or fine tuning in an existing algorithm can enhance its overall performance. CAEPSO has verified with an introduction of constriction factor that has massively improved its overall performance compared to AEPSO. However, if the adjustment is not appropriate then it can affect the overall performance as well as seen in CMPSO performance.
- Dynamic Parameterising Particle Swarm Optimisation (DPPSO) has given a fresh approach or angle to solve optimisation problems without worrying about the parameter setup. It is not only appropriate for any particular optimisation problem but it covers numerous type of optimisation problems.
- All algorithms including evolutionary algorithms are implemented in the online applications, in this case, the feasibility of mobile robot for path planning has

been proven. It is one of the approaches that can be used in the future for perhaps other available evolutionary algorithms.

- This research also focusing on more complex path planning task which is local path planning rather than global path planning. Global path planning requires a complete or partial knowledge of the environment layout while local path planning does not have that requirement which makes local path planning applicable for any layout without any prior procedures.

1.3 Research Question

1. What is the best swarm intelligence technique available and what is the advantages and disadvantages of these techniques [Chapter 2]?
2. What is the best classical technique in path planning for a mobile robot [Chapter 2]?
3. Is it possible to a PSO technique where each of its particles has its limitation based on their position on global and local best [Chapter 3]?
4. How can the dynamic approaches of PSO become flexible and adaptable to solve the different types of optimisation problems [Chapter 3]?
5. What is the performance of proposed techniques against existing evolutionary algorithms on standard benchmark functions [Chapter 4]?
6. What is the performance of proposed techniques compare to existing evolutionary methods on more complex optimisation problems such as engineering design optimisation problems [Chapter 5]?
7. Can all these evolutionary algorithms including proposed methods be implemented on a mobile robot for navigation optimisation problem [Chapter 6 and 7]?

8. Can evolutionary algorithms compete against classical path planning methods in order to optimise the navigation optimisation problems [Chapter 6 and 7]?

1.4 Research Objectives

1. To assess the influence of dynamic behaviour of particles in PSO toward the outcomes of the solutions.
2. To investigate the well-known swarm intelligence optimisation techniques available in the literature and study the advantages and disadvantages for each one of them.
3. To introduce the spring limit element into the PSO's algorithm velocity equation and assess the overall performance.
4. To introduce three new dynamic approaches of PSO algorithm and evaluate their performance.
5. To compare the proposed algorithms with another existing algorithm on several experiments and analyse the performance.
6. To conclude the best performing algorithm and the effect from the element introduced towards the PSO from the analysis done.

1.5 Research Motivation

Swarm Intelligence is a type of artificial intelligence based on the collective behaviour of decentralised and self-organised systems. The usage of the word *swarm* is not new and it was inspired by the social behaviour found in nature such as ant colonies, birds flocking, and fish schooling. Many researchers have studied these kinds of animals in order to understand how they communicate, evolve in nature, and achieve their goals. They concluded that such animals

have a decentralised control, lack of synchronisation, and simple identical members (Y. Liu & Passino, 2000).

Swarm intelligence characteristics within a group of non-intelligent individuals with decentralised control system can lead to the emergence of intelligent global collective behaviour that cannot be achieved by individuals. The most popular algorithm used in swarm intelligence is Evolutionary Algorithms (EA). There are many EAs that have been introduced since the early 60's, from Genetic Algorithm to the latest, Cuckoo Search Algorithm, it has been used in a variety of fields, for example in scheduling problem, data mining, networking problem, and also robotics. The most beneficial of EA is the algorithm that can initiate each agent to work, even those who only has a partial or limited knowledge about the environment because the information will be gathered through the agent's interaction with the environment.

1.6 Organisation of the Thesis

This thesis contains eight chapters and the summary of each chapter is as follows:

Chapter 1 covers the introduction and gives the general idea about the outline as well as the aims of this research.

Chapter 2 will be examining the literature on all the algorithms involved in this research in depth.

Chapter 3 aims to explain briefly about the new approaches introduced in this research.

Chapter 4 discusses the experiments designed in this research. It will also evaluate and assess the performance of five new approaches introduced in the previous chapter.

Chapter 5 discusses the results and analyses the performance of the Benchmark Functions experiment.

Chapter 6 is a continuation of the previous chapter, where the result and the analysis of the overall performance for all methods involved on the Engineering Design Problems will be discussed.

Chapter 7 deliberates on the experiments involving online applications, which is path planning using the mobile robot.

Chapter 8 will be recapping all the experiments that were carried out and discusses the overall performance for each of the algorithms involved.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This section discusses several Swarm Intelligence (SI) based algorithms by underlining the notable variations, the advantages and disadvantages, and the applications. These algorithms are Genetic Algorithms (GA), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), Differential Evolution (DE), and Cuckoo Search Algorithm (CSA). This section also discusses the classical path planning methods available in the literature. The algorithms involve Dijkstra's Algorithm (DA), Potential Field (PF), Rapidly-exploring Random Tree (RRT), and Probabilistic Road Map (PRM).

2.2 Swarm Intelligence

2.2.1 Particle Swarm Optimisation

The Particle Swarm Optimisation (PSO) is an optimisation technique which is quite popular and introduced by Kennedy and Eberhart in 1995 (Kennedy & Eberhart, 1995). It imitates swarm behaviour in birds flocking and fish schooling as a basic instrument to lead the particles to find the global optimal solutions. The PSO has three simple actions which are separation, alignment, and cohesion (illustrate in Figure 2-1) as explained by Del Valle et al., (2008). Separation behaviour is the ability to avoid the overcrowded local flockmates.

Alignment behaviour is the ability of moving in the path of the average track of local flockmates. Cohesion behaviour is the ability of moving towards the mean position of local flockmates (Valle, Venayagamoorthy, Mohagheghi, Hernandez, & Harley, 2008). The PSO algorithm's mathematical model comprises of three components which are previous particle's velocity component, cognitive component and social component and is written as follows (Kennedy & Eberhart, 1995; Valle et al., 2008):

$$Vi, j(t) = Vi, j(t - 1) + Ci, j + Si, j \qquad \textit{Equation 1}$$

$$Ci, j = c1 \times r1 \times (PBest_{i,j}(t - 1) - xi, j(t - 1)) \qquad \textit{Equation 2}$$

$$Si,j = c2 \times r2 \times (GBest_{i,j}(t - 1) - xi, j(t - 1)) \qquad \textit{Equation 3}$$

$$Xi, j(t) = Xi, j(t-1) + Vi, j(t) \qquad \textit{Equation 4}$$

where $V_{i,j}$ and $X_{i,j}$ are particle velocity and particle position respectively. t is the iteration number while i is the particle index. c_1 and c_2 represent the speed influence which regulates the length when flying towards the most optimal individual particle and the most optimal particles of the whole swarm. $PBest_{i,j}$ is the best position achieved so far by particle i and $GBest_{i,j}$ is the best position obtained by the neighbours of particle i. $r1$ and $r2$ are the random values (uniform distribution) between 0 and 1. The exploration behaviour is activated if either or both of the differences between the particle's best ($PBest_{i,j}$) and previous particle's position ($Xi, j(t)$), and between population's all-time best ($GBest_{i,j}$) and previous particle's position ($Xi, j(t)$) are massive, and the exploitation behaviour is activated when both of these values are small.

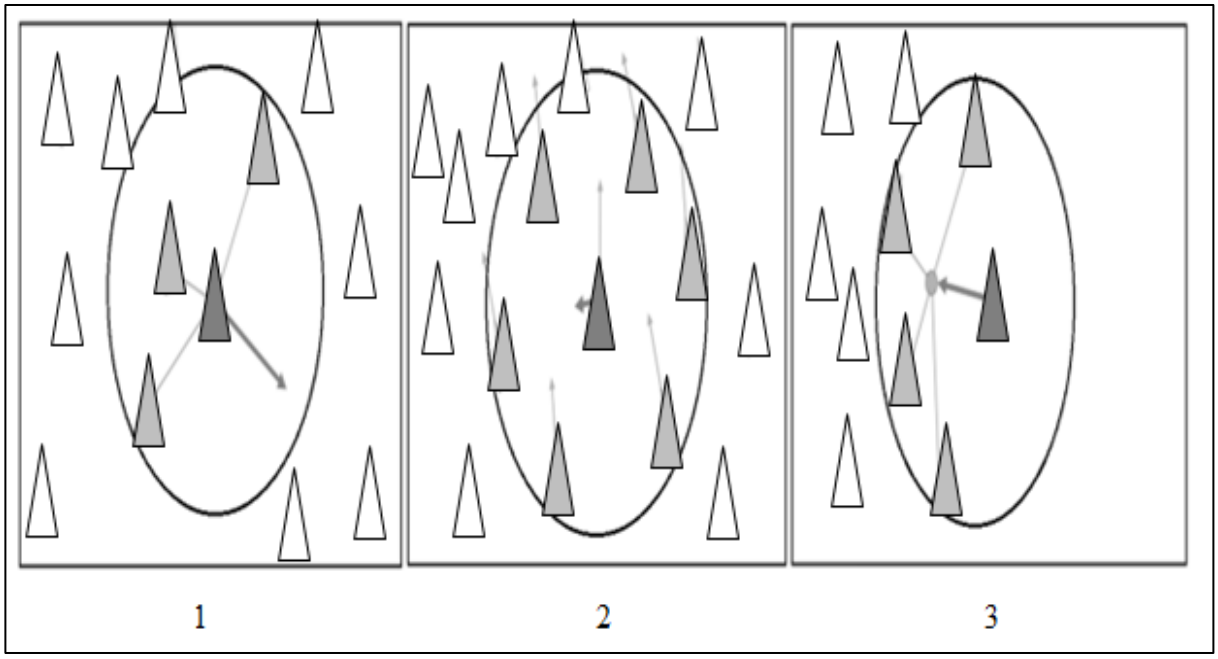


Figure 2-1. PSO Basic Behaviours. Figure 2-1-1 shows separation behaviour where particle avoiding other particles. Figure 2-1-2 shows alignment behaviour where particle moving towards the head of local flockmates and maintain the speed between them. Figure 2-1-3 shows cohesion behaviour where particle moving towards the average position of local flockmates (Valle et al., 2008).

PSO proves to be effective optimisation algorithm by searching an entire high-dimensional problem space. It is a vigorous stochastic optimisation technique based on the intelligence and movement of swarms. It implements the idea of social communication for problem solving and does not use the gradient of the problem being optimized. Hence, it does not need the optimisation problem to be different, as is essential by classic optimisation methods (Yan, Wu, Liu, & Huang, 2013). The optimisation of irregular problems with noise and changing over time can be determined using PSO (Gao, Kwong, Yang, & Cao, 2013; Kiranyaz, Ince, Yildirim, & Gabbouj, 2010; Senthil Arumugam, Ramana Murthy, Rao, & Loo, 2007). The parameters of PSO consist of position of agent in the solution space, number of particles, velocity and neighbourhood of agents (communication of topology). The PSO algorithm begins

by initialisation which is the same with almost all other optimisation techniques. The second step is measuring the fitness rate of each particle, then updating individual and global bests, and later, the velocity and the position of the particles also get updated. The second to fourth steps are repeated until the termination condition is met (Banks, Vincent, & Anyakoha, 2007, 2008; Senthil Arumugam et al., 2007).

Figure 2-2 shows the PSO algorithm behaviour over iterations. In the first iteration, all particles scattered within a search space to find the best solution (exploration). Each particle is assessed. Then, if the best new solutions are found, the personal and global best particles of that particular member of the swarm are updated. The convergence can be accomplished through drawing all particles towards the particle with the best solution. The PSO algorithm has many advantages. It is easy to implement, has only a few parameters to be configured, effective in global search, insensitive to scaling of design variables, and easily parallelized for concurrent processing (AlRashidi & El-Hawary, 2009; Imran, Hashim, & Khalid, 2013). However, PSO has the inclination to result in a fast and premature convergence in mid optimum points; in addition to having slow convergence in a refined search area (having weak local search ability) (AlRashidi & El-Hawary, 2009; Imran et al., 2013). PSO is applied in networking (Olascuaga-Cabrera, López-Mellado, & Mendez-Vazquez, 2011), power systems (Sa-ngawong & Ngamroo, 2015), signal processing (Iqbal, Zerguine, & Al-Dhahir, 2015), control system (Yudong Zhang, Wang, & Ji, 2015), machine learning (L. Wang et al., 2014), image processing (Uguz, Sahin, & Sahin, 2014), data mining (Fong, Wong, & Vasilakos, 2016), robotics (Alam & Rafique, 2015), and many more.

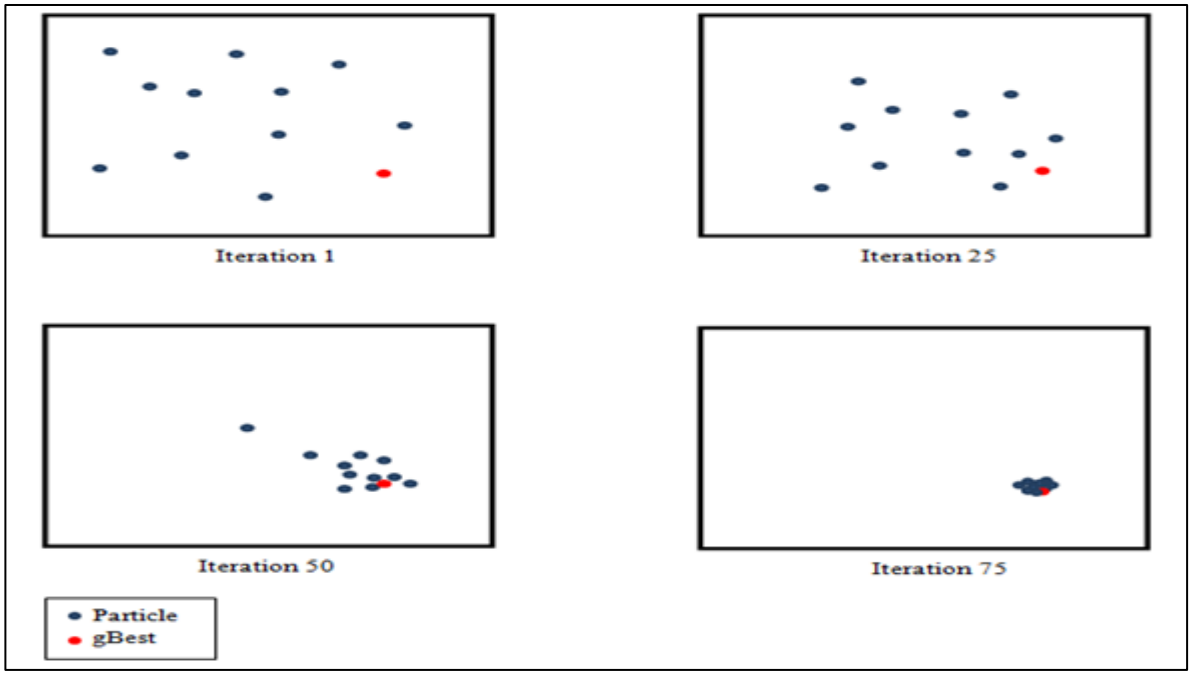


Figure 2-2. Particle Swarm Optimisation movement towards global optima over iteration numbers (Senthil Arumugam et al., 2007)

There are several studies have proven that they can enhance PSO in general. The number of the population size is one of the significant factors. Greater population size can increase the chances of quicker and precise convergence. Another method to enhance PSO is to find an equilibrium between exploration and exploitation action. High exploration at the beginning of iteration will give a higher chance for PSO to find a solution closer to global optima. When the PSO move towards the end of iteration, high exploitation would give a higher chance for particle to discover the most optimal solution within the promising area. A sub-swarm approach is another way that can be used to enhance the PSO performance which is quite regularly used nowadays. Assigning different objectives or tasks to each sub-swarm can also improve the competency of PSO in the multi-objective problems (Chang & Yu, 2013). Another approach to increasing the PSO performance is to set the donating components of the velocity equation

(dynamic velocity adjustment). Such approach can direct particles in altered directions and subsequently, faster convergence towards a global optimum can be achieved (A. Atyabi & Powers, 2013).

The two most noteworthy modifications in PSO are the introduction of inertia weight and constriction factors to velocity equation. Inertia weight (w) is proposed by Shi and Eberhart three years after PSO first introduction to control the influence of the previous velocity which also controls the exploration and the exploitation behaviours of the particle (Shi & Eberhart, 1998). If the w value is high then the step size is huge, consequentially, the occurrence in exploration behaviour. If the w value is small then the step size is small as well hence the exploitation behaviour take place. This new element has been recognised as the new standard form of velocity equation for basic PSO as illustrated in Equation 8:

$$Vi, j(t) = wVi, j(t - 1) + Ci, j + Si, j \qquad \textit{Equation 5}$$

The introduction of inertia weight, in general, has upgraded overall performance of PSO concerning the speed of convergence and the value of solutions. From there, much research has been done to find the finest configuration for inertia weight to enhance the convergence speed and the solutions' quality. Bratton and Kennedy (2007) recommend the implementation of an inertia weight value higher than 1.0 and declining eventually to a value lower than 1.0 with the aim of encouraging exploration at an early stage and exploitation within the best area found towards the end (Bratton & Kennedy, 2007). Clerc and Kennedy (2002) later propose the constriction factor, K to increase the opportunity of convergence and prevent particles from exiting the search space (Clerc & Kennedy, 2002).

$$Vi, j(t) = K[Vi, j(t - 1) + Ci, j + Si, j] \quad \textit{Equation 6}$$

Both of these variants have enhanced the overall performance of the PSO algorithm significantly. Eberhart and Shi have carried a research between these two variants and come to the conclusion that the constricted PSO perform better than the improved basic PSO (Eberhart & Shi, 2000). PSO can also be improved through their communication. Figueirdo and Ludermir (2012) have assessed five forms of communication topologies of global, local, von neuman, wheel, and four clusters. They concluded that global topology shows the most promising results compared to other topologies (Figueiredo & Ludermir, 2012). Bratton and Kennedy investigated the consequence of the number of particles in finding the solutions. Their research concludes that there is no exact number of population size that can be applied for all optimisation problems (Bratton & Kennedy, 2007).

2.2.2 Other Evolutionary Algorithms

John Holland introduced the Genetic Algorithm (GA) in 1975 as a search optimisation approach inspired from the natural selection process mechanism (Holland, 1975; M. Kumar, Husian, Upreti, & Gupta, 2010). The algorithm simulates the idea of the '*survival of the fittest*,' it mimics the processes in a natural system where the tough tends to adjust and live while the fragile usually perish. The population is ranked based on their solutions' fitness and a new member of the GA population is produced from several options of genetic operators such as crossover, reproduction, and mutation (Mitchell, 1995; Sivaraj & Ravichandran, 2011; Yan Cang Li, Li Na Zhao, & Zhou, 2011). The population is referred to as chromosomes (can also be illustrated in a set of strings). A new chromosome (a member of the population) in each

generation is formed using information extracted from the fittest chromosomes of the previous generation (Sivaraj & Ravichandran, 2011; Yan Cang Li et al., 2011).

GA has merits concerning only demanding limited parameter setting and initialising itself from many possible solutions rather than a single solution. The main downside of GA is the slow convergence towards the optimal values since the crossover and mutation processes are random (Meier, Gonter, & Kruse, 2013). GA has been implemented for various applications such as scheduling (Gupta, Kumar, & Agarwal, 2010; Y. Li & Chen, 2010; Lihong & Shengping, 2012; G. Zhang, Gao, & Shi, 2011), robotics (Abu-Dakka, Valero, & Mata, 2012; Zou, Ge, & Sun, 2012), machine learning (Busch et al., 2015; Kim, Jeong, McKay, Chon, & Joo, 2012; Litao, Tiejun, Xi, & Jin, 2012), signal processing (Chernbumroong, Cang, & Yu, 2015; Donglan, 2014), manufacturing (Deep & Singh, 2015; Jahanzaib, Masood, Nadeem, Akhtar, & Shahbaz, 2012; Kia, Khaksar-Haghani, Javadian, & Tavakkoli-Moghaddam, 2014), business (De Medeiros, 2015; Sirbiladze & Kapanadze, 2012; Yahya, Bae, Bae, & Kim, 2012), and many more.

Since its introduction, many researchers have conducted studies to enhance the performance of GA. There are numerous options especially for crossover and mutation operation to boost the value of solutions. For crossover operation, De Jong and Spears (1992) and Üçoluk (2002) introduce N-point crossover and segmented crossover which select some points for crossover rather than selecting only one crossover point (De Jong & Spears, 1992; Üçoluk, 2002). The main difference between them is N-point crossover is selecting a few breaking points randomly, while in the segmented crossover, only two breaking points are being used. As mention before, mutation operation is one of the most vital operators in GA since it is

able to push chromosomes in the direction of the improved solution. Hence, quite a lot of studies have been done to find different approaches for mutation operation.

Storn and Price introduced the Differential Evolution (DE) algorithm in 1997, a population-based algorithm which is similar to GA in term of employs similar operators; crossover, mutation, and selection. In term of generating new solutions, DE depends on mutation operation while GA depends on crossover operation as their mechanism (Storn & Price, 1997). DE uses the mutation process as the core search instrument and utilises the selection process as the mechanism to direct the search towards the promising areas in the search environment. DE utilises three properties for generating a new population iteratively which are Target Vector, Mutant Vector, and Trail Vector. The target vector is the vector which consists the solution for the search area. The mutant vector is the mutated target vector with the trail vector is the outcome vector after crossover process between the target vector and the mutant vector. As mention before, the basic procedures of the DE algorithm are similar to GA with slight difference on the order of the routines.

The first step for DE algorithm is initialisation of the population using either random distribution or heuristic-based distribution, followed by valuation of each population's member to determine their fitness. Then, new vectors are generated by adding the weighted difference of two members of the population with the third vector which known as mutation. During the crossover process, the vectors are blended up and after that, the algorithm takes the last process of selection (Das, Nagaratnam Suganthan, & Member, 2011; Price, Storn, & Lampinen, 2005). However, DE has a slight shortage regarding slow convergence and being unstable (Y. Wu, Lee, & Chien, 2011). DE is applied in countless applications such as electrical engineering (Qing, 2009), image processing (Pei, Zhao, & Liu, 2009), machine learning (W.-A. Yang, Zhou,

& Tsui, 2015), robotics (Al-Dabbagh, Kinsheel, Mekhilef, Baba, & Shamshirband, 2014), and economy (L. Wu, Wang, Yuan, & Chen, 2011).

In general, DE overall performance can be enhanced by adding more members of the population. Harmonising between exploration and exploitation behaviour where the scaling factor (controls the step size) is high at the beginning and getting lower towards the end of an iteration. The introduction of elitism can be another step to improve DE performance where elitism can prevent the best solution from being destroyed when the new generation is generated. There is countless modified version of DE invented since its introduction by Storn and Price. Mezura-Montes et al. (2006) have discussed some variants of DE and make a comparative research between them (Mezura-Montes, Velázquez-Reyes, & Coello Coello, 2006). The variants discoursed are *DE/rand/1/bin*, *DE/rand/1/exp*, *DE/best/1/bin*, *DE/best/1/exp*, *DE/current-to-best/1*, *DE/current-to-rand/1*, *DE/current-to-rand/1/bin*, and *DE/rand/2/dir*. The main differences between them are regarding individuals' selection for mutation, the numbers of solutions selected and the type of crossover (Das & Suganthan, 2011).

The Cuckoo Search Algorithm (CSA) is one of the newest metaheuristic approaches proposed by Yang and Deb in 2009 (X. S. Yang & Deb, 2009). This algorithm is inspired by the activities of cuckoo species, such as brood parasites, and the characteristics of Lévy flights, such as fruit flies and some birds. CSA uses three fundamental rules or operations in its application. Each cuckoo is only permitted to lay one egg in each iteration. The cuckoo selects the nest randomly to lay its egg. After that, all eggs and nests are evaluated with only eggs and nests with the high value of fitness are retained for the next generation. Then, with a fixed number of available host nests, the egg laid by a cuckoo is discovered by a host bird using probability $p_a \in [0, 1]$. If the host nests discovered the egg laid, the host has options either to

throw the egg away or abandon the nest and completely build a new nest. The last supposition can be approximated as a fraction, p_a of the total n nests that are substituted by new nests with a new random solution. The algorithm also can be stretched to a more complex point where each nest comprises many eggs (X. S. Yang & Deb, 2009; X.-S. Yang & Deb, 2010).

CSA is brilliant with multimodal objective functions as it only requires fewer numbers of parameters to be fine-tuned compared to other approaches. It has an oblivious convergence rate to the parameter p_a where on some occasions fine tuning the parameters is not necessary (X. S. Yang & Deb, 2009, 2013; X.-S. Yang & Deb, 2010). CSA is implemented to various fields including neural network (Chaowanawatee & Heednacram, 2012), embedded systems (A. Kumar & Chakarverty, 2011), signal processing (Araghi, Khosravi, & Creighton, 2015), economics (Basu & Chowdhury, 2013), business (X. S. Yang, Deb, Karamanoglu, & He, 2012), and TSP problem (Ouyang, Zhou, Luo, & Chen, 2013).

Walton et al. (2011) have introduced a variant for CSA entitled Modified Cuckoo Search (MCS) where their key aim is to increase the convergence speed (Walton, Hassan, Morgan, & Brown, 2011). This enhancement includes an extra step in which the top eggs do some information input. They have tested MCS on a number of benchmark functions, and the results show that MCS managed to outperform the standard CSA. Another important variant for CSA is Quantum Inspired Cuckoo Search Algorithm (QICSA) proposed by Layeb in 2011 as well (Layeb, 2011). The author combined few elements from quantum computing principles like qubit representation, measure operation, and quantum mutation onto cuckoo search algorithm. The core objectives are to boost the diversity and the performance of regular CSA. The results show that there are still some weaknesses in QICSA and the author recommended to integrate

a local search and parallel machines to enhance the efficiency and upsurge of the convergence speed (Layeb, 2011).

2.2.3 Homogenous and Heterogeneous Swarm

Homogenous swarm is defined as a similar or same type of agents used in the whole swarm, and it is a quite common approach. Heterogeneous Swarm, on the other hand, is define as a unique or different type of agents consist in the whole swarm. Since homogeneous swarm is similar, the same sort of behaviour is expected from all agents while in heterogeneous, different type of behaviours between agents are anticipated.

There are many studies done using homogenous swarm. The details of these approaches can be found from here (Martinoli & Easton, 2003). The same thing goes to heterogeneous swarm where many scholars start to exploit this type of approach to see the differences between these two types of a swarm. The details of the heterogeneous swarm can be found from the same research (Martinoli & Easton, 2003).

2.2.4 Macroscopic and Microscopic Modelling

Microscopic modelling is a mechanism that focuses on individual robots in its modelling, where members of the swarm (chromosome, particle or vector) represent the robots in the fleet (Lerman, Martinoli, & Galstyan, 2005)(Lan & Li, 2010). Since in microscopic modelling, the behaviour of each robot is modelled explicitly, a large number of robots is required for this modelling mechanism, besides of requiring plenty of computational processes

which might be difficult to model in the real-world. Hence, simulations are the most common tools used to analyse and validate microscopic models for swarm robotics systems.

Macroscopic modelling is a mechanism that considers swarm robotics systems as a whole. This mean, in this modelling mechanism, each robot represents a swarm of multiple possible actions resulting in a reduction in swarm size and consequently the necessary computation (Martinoli & Easton, 2003)(Lerman et al., 2005)(Lan & Li, 2010). Lerman et al. (2005) have investigated the feasibility of macroscopic modelling in a group of reactive robots with a focus on the dynamics of group behaviours and issues related to robots' collaborations. Lerman et al. (2005) also provided a review of methods used for macroscopic modelling and analyses of the collective behaviour of the swarm robotic systems.

2.3 Motion Planning: Classical Path Planning Algorithm

The motion planning and navigation approaches can be categorised into two broad classes of heuristic-based and classical methods. Atyabi and Powers (Adham Atyabi & Powers, 2013) consider approaches such as Cell Decomposition (CD), Potential Field (PF), Road Map, and Subgoal Network in the classical methods category. The main disadvantages of these methods are that they are computationally intensive and incapable of handling uncertainty. In addition, in their basic forms they are mostly unreliable in dynamic and/or partially known environments. The majority of the classical approaches are dependent on having extensive prior knowledge of the environment in order to generate the accessible path from the starting location towards the goal destination without which the optimal motion planning is not feasible. Heuristic-based methods, on the other hand, can overcome the problems faced by classical methods, as they are capable of handling the unknown or partially known environment. It is

due to their ability to generate a set of a temporary plan within each iteration of their algorithms in a way to bring them closer to the destination location. Meanwhile, in evolutionary-based methods, rather than generating an overall plan from the prior knowledge (e.g. environment map) and executing it, in each step, a sub-population of possible manoeuvers are considered, amongst which the one that best addresses both the robot and overall mission (e.g. reaching to a destination) is then executed. Atyabi and Powers (2013) include approaches such as Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), Neural Networks (NN), and Ant Colony Optimisation (ACO) as heuristic-based methods for navigation (Adham Atyabi & Powers, 2013). These approaches are briefly explained in the following sections.

The classical methods utilised in this research are Potential Field (PF), Dijkstra's Algorithm (DA), Rapidly-explore Random Tree (RRT), and Probabilistic Road Map (PRM). These four methods are selected for being well known in the community and being widely and successfully applied in path planning problems (Chen, Yu, Su, & Luo, 2014; Elbanhawi & Simic, 2014; Norouzi, De Bruijn, & Miró, 2012; H. Wang, Yu, & Yuan, 2011). The performance of these methods are proven to be quite competitive and decent in global path planning. However, PF, RRT, and PRM have also been applied for local path planning (Chen et al., 2014; Contreras-Cruz, Ayala-Ramirez, & Hernandez-Belmonte, 2015; Norouzi et al., 2012). For heuristic-based methods utilised, the selection of the algorithms -GA, DE, variants of PSOs, and CSA- is based on their popularity amongst researchers and their potential in solving the local path planning problem (Adham Atyabi & Powers, 2010; Chiu, Cheng, & Chang, 2012; Roberge, Tarbouchi, & Labonte, 2013; Zou et al., 2012). The details of these methods are discussed in the following sections.

2.3.1 Potential Field (PF)

In Potential Field (PF), the differences between two opposite forces (e.g., attraction and repulsion) are utilised for manoeuvring a robot in the environment (Khatib, 1986). The robot ranks its next manoeuvring options at each location based on how close they are to the locations of known obstacles and the final destination (repulsion and attraction forces respectively) (Barraquand, Langlois, & Latombe, 1992; Hwang & Ahuja, 1992; Y. Wang & Chirikjian, 2000). From the time that it was introduced by Khatib in 1986 (Khatib, 1986), Potential Field has been widely utilised in the robotics community and several variations. Modified versions of it have been introduced to address its shortcomings, such as being computationally intensive and trapping in local minima (Rimon & Koditschek, 1992; Seda, 2007; Y. Wang & Chirikjian, 2000). A decentralized PF-based controller is utilised by Song and Kumar (2002) for manoeuvring and directing teams of robots in a scenario in which robots are tasked to approach targets as a team, trap them and lead them towards a destination location (Song & Kumar, 2002). Cheng and Zelinsky (2005) tackled the local minima trapping problem of PF by suggesting the usage of high magnitude temporary attraction forces towards random positions when robots are trapped among obstacles (Cheng & Zelinsky, 1995).

Sfeir et al. (2011) explored the impact of repelling force on PF, aiming to reduce oscillations and avoiding a collision when the target is close to obstacles (Sfeir, Saad, & Saliah-Hassane, 2011). Other proposed approaches to address the local minima problem of the PF include injecting noise or randomness in the local minima, tracking back behaviour, and introducing a tangential repulsive force field around obstacles (Siegwart & Nourbakhsh, 2004).

2.3.2 Dijkstra's Algorithm (DA)

Dijkstra's Algorithm (DA) is introduced by Edsger Dijkstra in 1959 (Dijkstra, 1959; H. Wang et al., 2011; Yin & Wang, 2010) and consider as a classical algorithm that proved its efficiency in finding the shortest path within a web of inter-connected nodes that represent manoeuvrable spaces among obstacles. The main disadvantages of the approach are being computationally intensive and having a poor search efficiency if the distance between the starting point and destination point is massive (B. L. B. Liu et al., 1994; Noto & Sato, 2000). Noto and Soto (2000) have proposed an extended version of DA to overcome the problem (Noto & Sato, 2000). DA is often used in routing problems (Dijkstra, 1959; H. Wang et al., 2011; Yin & Wang, 2010).

2.3.3 Rapidly-Exploring Random Tree (RRT)

The Rapidly-Exploring Random Tree (RRT) is based on stochastic search strategies which were introduced by LaValle and Kuffner Jr. (S M LaValle, 1998; Steven M. LaValle, 2011). The RRT is applicable for single query problems. The RRT approach to path planning is to construct a tree which revolves around random points picked in the searching environment. The start point is the root node for the RRT, once a random point is selected, then the closest node constructs an incremental distance from itself towards a selected random point. At the end of an additional distance is a new node. This process is repeated until the goal point is found and will form a path which looks like a tree and cover almost all the empty spaces in the search environment (Elbanhawi & Simic, 2014; Jaillet & Porta, 2013; Kuffner & LaValle, 2000). In this paper, the variation of RRT named RRT-based local path planning is used as one of the comparison techniques. The same concept and approach of original RRT are used, the only

difference is it does not construct a path before it is moving. A random point or target point is chosen while it is moving depending on some probability (Tian et al., 2007).

2.3.4 Probabilistic Road Map (PRM)

The Probabilistic Road Map (PRM) differs from the RRT in the sense that it is applicable for multi-query planning. The PRM constructs a path between the start point and target point by connecting random nodes in the free spaces within the environment. The algorithm starts with scattered random points within the environment. The random points which lay on the obstacle are neglected, and only the random points on free spaces are considered as nodes. All these nodes are linked to each other to determine their feasibility. Once all nodes are connected, the links which intersect with an obstacle are considered as not feasible and are neglected. In the next step of the algorithm, the possible paths from starting point to end point are gathered from the existing routes and all infeasible solutions get omitted. The shortest path between the starting point and destination is usually chosen as the final path (Elbanhawi & Simic, 2014; Geraerts & Overmars, 2004; Kavraki et al., 1996).

2.4 Path Planning using Mobile Robot

This subsection discusses several past studies related to path planning for a mobile robot. Chia et al. (2010) implement the Ant Colony Optimisation (ACO) algorithm to mobile robot system to solve mobile robot path planning problem. They use simulation platform to assess the performance of the proposed algorithm. The simulated results show that ACO can find the possible path for mobile robot, moving from the start position (nest) to the target location (food) with a collision-free manoeuvre (Chia, Su, Guo, & Chung, 2010). Zhang et al.

(2013) propose a multi-objective path planning algorithm based on particle swarm optimisation for robot navigation in such various danger environment that robots must evade, such as a fire in a rescue mission, landmines and enemies in war field (Yong Zhang, Gong, & Zhang, 2013).

Contreras-Cruz et al. (2015) propose an approach combines the Artificial Bee Colony (ABC) algorithm as a local search operator and the Evolutionary Programming (EP) to cultivate the possible path found by a set of local operators. They have tested the algorithm in several scenarios in simulation platform. They also implemented the approach on Pioneer 3-AT for online application (Contreras-Cruz et al., 2015).

2.5 Robot Operating System (ROS)

Robot Operating System (ROS) is developed at the Stanford University for the Stanford Artificial Intelligence Robot Project (STAIR) in 2007. The first official version was launched in 2010. ROS is an environment system to develop robot applications with a repository of open software sources. ROS is intended to support the key functions in Robotics, which are visual odometry, planning and control, object recognition, mapping and navigation. Willow Garage fully maintained ROS at this moment and most of its packages and updates are coming from all over the world via ROS communities. ROS used Ubuntu Linux as its main platform with MacOS, Linux, and Windows still in the experimental stage. Most of ROS codes are developed in C++ and Python languages but ROS has a multi-lingual support feature that is adaptable to other programming languages.

The field of robotics is a vast world and serves a variety of applications in the area of science and technology in academia and industries. There are many different types of robots in

operation in such diverse environments that it is often difficult to replace a robot with a different one. In such circumstance, there would be a lot of duplication of effort for each robot to achieve the more basic tasks first. A single framework is not possible for all the application of robotics. However, a single framework can be used to achieve the core and the more common tasks so that researchers can work on the more complex and advanced projects. Such a single framework will have a huge demand for memory and processing power, which may not be possible for smaller robots. All the processing tasks may be transferred from mobile robotic platform to a centralised control system, which can now control not just a single robot but also a swarm of robots. However, this would add a requirement of a strong networking of booth peer-to-peer and server-client types in the unified framework.

2.6 Summary

This chapter merely focuses on literature research where it discusses well-known swarm intelligence from Genetic Algorithm to Cuckoo Search Algorithm (CSA). It also briefly discusses the types of swarm intelligence which are homogeneous and heterogeneous. The forms of implementation for swarm intelligence approaches are also being discoursed in this chapter. This chapter also discusses several classical motion planning methods and case studies involved in path planning using a mobile robot. In the following chapter, the specifics discussion about the introduction of new approaches based on spring limitation and the dynamic behaviour of heterogeneous swarm are going to be explored.

CHAPTER 3

METHODOLOGIES

3.1 Introduction

In the previous chapter, comprehensive literature studies are done on several topics including evolutionary algorithms, classical methods for path planning and path planning using a mobile robot. This chapter will be focusing on the details of five proposed methods and how they are created and inspired from. The first two approaches are based on the spring physical limitation. The remaining three methods are based on the dynamic behaviour of heterogeneous swarm.

3.2 Morphology Particle Swarm Optimisation

Let's recap the discussion on PSO in the previous chapter where each particles of PSO position in the search space is represented by a position; *X*. PSO evolves its solutions towards better regions of the search space by updating the particles' position in the search space using a velocity, *V*. The best solution found by each particle is referred to as personal best (*PBest*) and the best performing particle found in the whole swarm is referred to as global best (*GBest*). In PSO, particles update their velocities and positions in the search space using the following equations:

$$Vi, j(t) = wVi, j(t - 1) + Ci, j + Si, j \qquad \textit{Equation 7}$$

$$Ci, j = c1 \times r_1 \times (PBesti,j(t - 1) – xi, j(t - 1)) \quad \text{Equation 8}$$

$$Si,j = c2 \times r_2 \times (GBesti, j(t - 1) – xi, j(t - 1)) \quad \text{Equation 9}$$

In Equation 16, *Vi, j(t)* represents the velocity in iteration *t*. *i* and *j* represent the particle's index and the dimension in the search space respectively. *c1* and *c2* represent the acceleration coefficients of cognitive (*Ci, j*) and social (*Si, j*) components respectively. $r_{1,j}$ and $r_{2,j}$ are random values in the range of [0,1] while *w* is the inertia weight that controls the influence of the last velocity in the updated version.

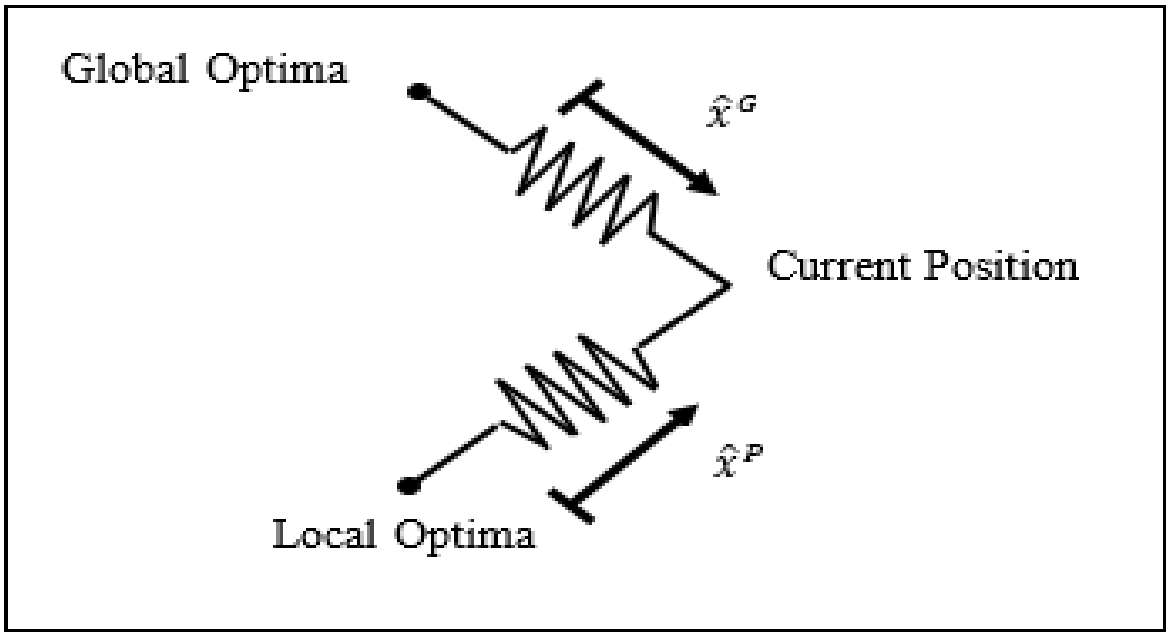


Figure 3-1. Morphology PSO.

The original PSO proposed is unstable, and particles have the tendency to move away from the attractor (Langeveld, 2011; Tuppadung & Kurutach, 2011). Therefore, in late 1998,

Shi and Eberhart (1998) proposed an improvement for the first PSO introduced in 1995. The authors proposed an introduction of inertia weight, *w* parameter to the Equation 16.

In this research, the PSO equation is derived from the spring equation. Since the spring equation has a limit, the equation derived will also have a limit. Once the equation has a limit, it becomes more realistic for online application, especially in mobile robot application. The equation below is derived based on Hooke's law:

$$F_i = k^G(x_i^G - x_i) - k^P(x_i^P - x_i) \qquad \textit{Equation 10}$$

The force of G_{best} and P_{best} applied towards the particle is given by Equation 19 as shown in Figure 3-1. Force is equal to mass times acceleration. Hence, Equation 19 becomes Equation 20 where the randomness is contributed by the noise in the system.

$$ma_i = -k^G r[0,1](x_i^G - x_i) - k^P r[0,1](x_i^P - x_i) \qquad \textit{Equation 11}$$

$$m\left(\frac{V_i(t+\Delta t) - V_i(t)}{\Delta t}\right) = -k^G r(x_i^G(t) - x_i(t)) - k^P r(x_i^P(t) - x_i(t)) \qquad \textit{Equation 12}$$

After that, acceleration is converted to velocity, as acceleration is equal to difference of velocity over time.

$$m\big(V_i(t+1) - V_i(t)\big) = -k^G r(x_i^G(t) - x_i(t)) - k^P r(x_i^P(t) - x_i(t)) \qquad \textit{Equation 13}$$

$$V_i(t+1) - V_i(t) = c^G r(x_i^G(t) - x_i(t)) + c^P r(x_i^P(t) - x_i(t)) \qquad \textit{Equation 14}$$

Then, constant *m* is brought to the right hand side of the equation and becomes $\frac{k}{m}$. Since both are a constant, they will be replaced by only one constant *c*, shown in Equation 23.

$$V_i(t+1) = c^G r\left(x_i^G(t) - x_i(t)\right) + c^P r(x_i^P(t) - x_i(t) + V_i(t) \qquad \textit{Equation 15}$$

Hence, Equation 24 has a similarity with the PSO in Equation 16. Equation 25 gives the graph's plot function:

$$f(x) = xe^{-ax} \qquad \textit{Equation 16}$$

Then, the function from Equation 25 is added to the Equation 16 and becomes Equation 26 as shown:

$$V_i(t+1) = wV_i(t) + c_1 r\big(P_{best} - x_i(t)\big)e^{-a_P\left(P_{best} - x_i(t)\right)} + c_2 r\big(G_{best} - x_i(t)\big)e^{-a_G\left(G_{best} - x_i(t)\right)} \qquad \textit{Equation 17}$$

$$V_i(t+\Delta t) = \frac{x_i(t+\Delta t) - x_i(t)}{\Delta t} \qquad \textit{Equation 18}$$

$$x_i(t+1) = x_i(t) + V_i(t+1) \qquad \textit{Equation 19}$$

Since CPSO performs quite exceptionally in several studies mentioned in the previous chapter (Eberhart & Shi, 2000; Clerc & Kennedy, 2002), the constriction factor, *K*, is also considered to be introduced in the Morphology PSO (MPSO) equation. Hence, the equation becomes as follow and named as Constricted Morphology PSO (CMPSO):

$$V_i(t+1) = K(V_i(t) + c_1 r\big(P_{best} - x_i(t)\big)e^{-a_P\left(P_{best} - x_i(t)\right)} + c_2 r\big(G_{best} - x_i(t)\big)e^{-a_G\left(G_{best} - x_i(t)\right)}) \qquad \textit{Equation 20}$$

$$x_i(t+1) = x_i(t) + V_i(t+1) \qquad \textit{Equation 21}$$

3.3 Dynamic Approaches of Particle Swarm Optimisation

This proposed PSO is inspired by two main approaches which are heterogeneous swarm and sub-swarm approach (Cesmeci & Gullu, 2010; Xu, Chen, & Yu, 2006; J. Zhang, De-Shuang, & Kun-Hong, 2007). Heterogeneous Swarm approach has proven to be a better performer compared to homogenous swarm (A. P. Engelbrecht, 2011; Andries P Engelbrecht, 2010). The aim for this proposed PSO is to remove or reduce the parameter setting in PSO. Only several parameter settings are required for PSO, which are inertia weight and acceleration coefficients, but different parameter settings may be needed for different problems. The combination of heterogeneous parameter setting swarm in sub-swarm gives each particle to have several options to select the proper parameter at their current positon. The approach used for this paper is where each swarm will choose its fittest sub-swarm and use the sub-swarm's parameter setting as its own. Therefore, each swarm will be using different parameter setting over the iterations depending on its current fitness.

3.3.1 Dynamic Parameterisation Particle Swarm Optimisation (DPPSO)

The parameter setting options proposed are the combination of three parameter settings for inertia weight (w) and three parameter settings for acceleration coefficients ($c1$ and $c2$). For inertia weight, the parameter settings used are fixed inertia weight (FIW), random inertia weight (RANDIW) and linear decreasing inertia weight (LDIW) (Rapaić & Kanović, 2009; Ratnaweera, Halgamuge, & Watson, 2004). For acceleration coefficients, the parameter settings used are fixed acceleration factors (FAC), random acceleration factors (RANDAC) and time varying acceleration factors (TVAC) (Rapaić & Kanović, 2009; Ratnaweera et al., 2004).

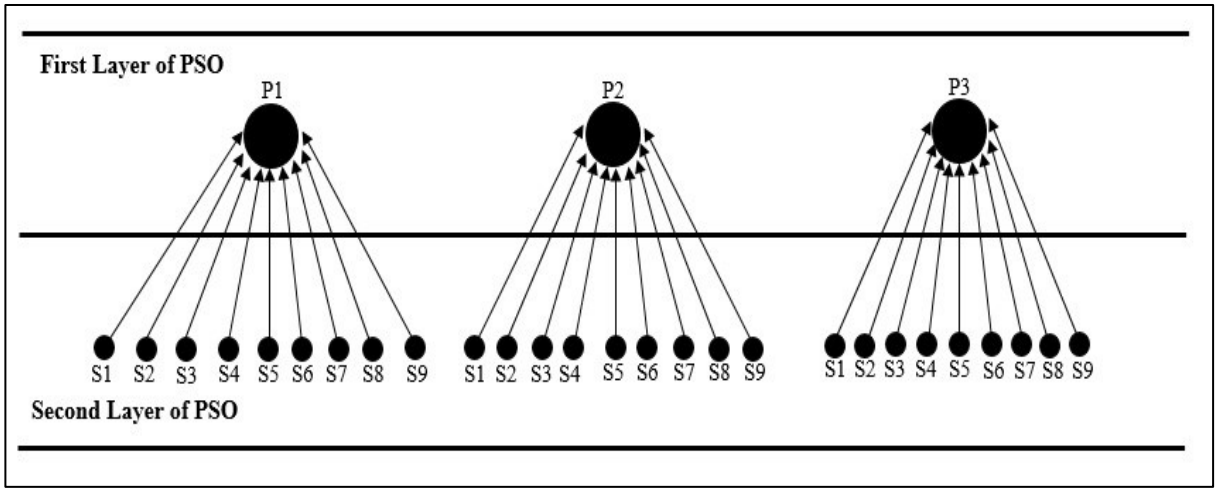


Figure 3-2. The Structure of Dynamic Parameterising PSO (DPPSO).

This method consists of two layers where the first layer consists the particle from the swarm and the second layer consists of nine different configuration sub-swarm as illustrated in Figure 3-2. The idea behind this proposed method is each particle in the first layer is using the best parameter settings based on its current position. The best parameter settings are determined by its sub-swarm in the second tier. The sub-swarm consists of nine possible parameters configuration where all of them will be tested against a fitness function. The sub-swarm which has the fittest value will be selected as the configurations for the particle in the first layer. The combination of these parameter settings produces the following nine sub-swarms for each swarm:

- Sub-swarm 1: FIW + FAC

$$V_i(t+1) = 0.7299V_i + 0.5r\left(P_{best} - x_i(t)\right) + 2.5r\left(G_{best} - x_i(t)\right) \quad \textit{Equation 22}$$

The fixed value of Inertia Weight (w) and Acceleration Coefficients for Cognitive and Social Components (c_1 and c_2) are used which is 0.7299 for w, 0.5 for c_1 and 2.5 for c_2.

- Sub-swarm 2: FIW + RANDAC

$$V_i(t+1) = 0.7299V_i + ((2.5-0.5)(1-r)+0.5)r\big(P_{best} - x_i(t)\big) + ((2.5-0.5)\,(1-r)+0.5)r\big(G_{best} - x_i(t)\big) \quad \textit{Equation 23}$$

The value for Inertia Weight, *w* is fixed with 0.7299 with the values of acceleration coefficients for both components are randomised between 0.5 and 2.5.

- Sub-swarm 3: FIW + TVAC

$$V_i(t+1) = 0.7299V_i + (\frac{c_i - c_f}{t_{max}}t + c_f)r\big(P_{best} - x_i(t)\big) + (\frac{c_f - c_i}{t_{max}}t + c_i)r\big(G_{best} - x_i(t)\big) \quad \textit{Equation 24}$$

The value for Inertia Weight (*w*) is fixed for this sub-swarm which is 0.7299 and the acceleration coefficients used the Time-Varying option where for c_1, the value is decreasing from 2.5 to 0.5 over the iteration numbers while c_2 value is increasing from 0.5 to 2.5 over time.

- Sub-swarm 4: RANDIW + FAC

$$V_i(t+1) = (0.5 + \frac{r}{2})V_i + 0.5r\big(P_{best} - x_i(t)\big) + 2.5r\big(G_{best} - x_i(t)\big)$$

The value for inertia weight (*w*) is randomly selected between 0.5 and 1.0 with the fixed acceleration coefficients (c_1 and c_2) are used which are 0.5 and 2.5 respectively.

- Sub-swarm 5: RANDIW + RANDAC

$$V_i(t+1) = (0.5 + \frac{r}{2})V_i + ((2.5-0.5)(1-r)+0.5)r\big(P_{best} - x_i(t)\big) + ((2.5-0.5)(1-r)+0.5)r\big(G_{best} - x_i(t)\big)$$

A random value between 0.5 and 1.0 is used for inertia weight (*w*) with both acceleration coefficients (c_1 and c_2) also used a random value between 0.5 and 2.5 for every iteration.

- Sub-swarm 6: RANDIW + TVAC

$$V_i(t+1) = \left(0.5 + \frac{r}{2}\right)V_i + \left(\frac{c_i - c_f}{t_{max}}t + c_f\right)r\left(P_{best} - x_i(t)\right) + \left(\frac{c_f - c_i}{t_{max}}t + c_i\right)r\left(G_{best} - x_i(t)\right)$$

The inertia weight, *w* is randomly picked between 0.5 and 1.0. The acceleration coefficients for social and cognitive components are vary depending on the time where the c_1 value is decreasing from 2.5 to 0.5 while the c_2 value is increasing in the opposite direction.

Sub-swarm 7: LDIW + FAC

$$V_i(t+1) = \left(\frac{w_f - w_i}{t_{max}}t + w_i\right)V_i + 0.5r\left(P_{best} - x_i(t)\right) + 2.5r\left(G_{best} - x_i(t)\right)$$

The linear decreasing approach for inertia weight (*w*) is used for this sub-swarm where initially the value of *w* is 0.9 and eventually decreasing to the final value of 0.4. Meanwhile, the value of the acceleration coefficients remains throughout the iteration with 0.5 for c_1 and 2.5 for c_2.

- Sub-swarm 8: LDIW + RANDAC

$$V_i(t+1) = \left(\frac{w_f - w_i}{t_{max}}t + w_i\right)V_i + ((2.5-0.5)(1-r) + 0.5)r\left(P_{best} - x_i(t)\right) + \left((2.5-0.5)\frac{(1-r)}{1} + 0.5\right)\left(G_{best} - x_i(t)\right)$$

This sub-swarm combines linear decreasing method for inertia weight with random values of acceleration coefficients. The inertia weight (*w*) value set 0.9 at the beginning and declining towards 0.4 in the end. The acceleration coefficients (c_1 and c_2) for both components are randomly selected between 0.5 and 2.5.

- Sub-swarm 9: LDIW + TVAC

$$V_i(t+1) = \left((w_f - w_i)\frac{(t_{max} - t)}{t_{max}} + w_i\right)V_i + \left(\frac{c_i - c_f}{t_{max}}t + c_f\right)r\left(P_{best} - x_i(t)\right) + \left(\frac{c_f - c_i}{t_{max}}t + c_i\right)r\left(G_{best} - x_i(t)\right)$$

This sub-swarm is the only sub-swarm where all components are related to the iteration numbers. The inertia weight (*w*) decreasing over time from 0.9 to 0.4 and the acceleration

coefficient for cognitive component (c_1) decreasing over time from 2.5 to 0.5. Meanwhile, the value of the acceleration coefficient for social component (c_2) is increasing from 0.5 to 2.5 over time.

where:

$$r = [0,1], c_i = 0.5, c_f = 2.5, w_i = 0.9, w_f = 0.4, t = iteration$$

The selection criteria used, as mentioned before, is the fittest sub-swarm will be chosen as the main swarm. This step is crucial to ensure that the swarm is using the right exploration and exploitation behaviour depending on its current position.

3.3.2 Dynamic Acceleration Coefficients Particle Swarm Optimisation (DACPSO)

Acceleration Coefficient is one of the essential components in PSO equation. It controls the influence of social and cognitive component of PSO. The balance between social and cognitive component could determine the overall performance of PSO in the optimisation problem and influence the outcome. If the value set for acceleration coefficients is not appropriate for the problem, the performance of PSO might drop. However, there is no one absolute value of acceleration coefficients that can be used on all optimisation problems. There are two important approaches introduced in an attempt to address this problem, which is time-varying acceleration coefficient (TVAC PSO) and random acceleration coefficients (RANDAC PSO) (Rapaić & Kanović, 2009; Ratnaweera et al., 2004). However, these methods still cannot address all optimisation problems. Therefore, the idea of using dynamic acceleration coefficients approach comes into play as it aims to change the acceleration coefficients

depending on the search space and then independently adapts to the optimisation problem thus overcoming it.

- First Sub-group

$$V_i(t+1) = w_i V_i + 2.5r\big(P_{best} - x_i(t)\big) + 0.5r\big(G_{best} - x_i(t)\big)$$

The influence of cognitive component is considered higher than the influence of social component which means the mean direction of the particle is more towards the personal best rather than global best. The value set for the cognitive component is 2.5 while the value set for the social component is 0.5. The linear decreasing inertia weight approach is used for this proposed method.

- Second Sub-group

$$V_i(t+1) = w_i V_i + 1.5r\big(P_{best} - x_i(t)\big) + 1.5r\big(G_{best} - x_i(t)\big)$$

For the second sub-group of particles, the influence of cognitive and social components are equal. This option will make sure every particle belongs to this sub-group is not leaning towards any components. The value set for both components is 1.5 and the linear decreasing inertia weight approach is also applied for this sub-group.

- Third Sub-group

$$V_i(t+1) = w_i V_i + 0.5r\big(P_{best} - x_i(t)\big) + 2.5r\big(G_{best} - x_i(t)\big)$$

This sub-group is using the standard PSO approach where the social component has greater influence compare to the cognitive component. This specification procedure aims to direct the

particles toward the global best from the very beginning and make the convergence faster. The value set for the cognitive component is 0.5 and 2.5 for the social element.

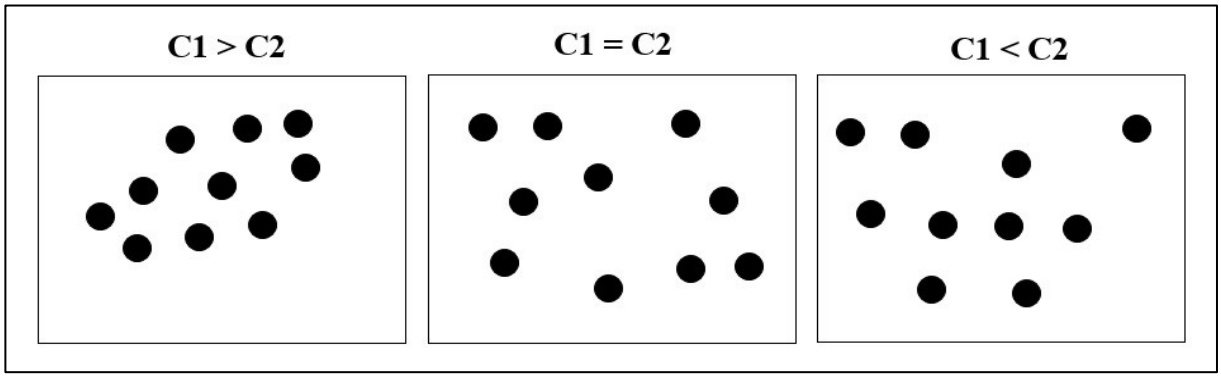


Figure 3-3. Dynamic Acceleration Coefficients PSO (DACPSO) Population Size Initially.

The details on how DACPSO work are illustrated in Figure 3-3, Figure 3-4, and **Error! Reference source not found.**. In Figure 3-3, the whole population is divided into three sub-groups with three different configuration for acceleration coefficients (*c1* and *c2*). In the first group, the value of acceleration coefficient for the cognitive component is greater than the value of acceleration coefficient for social element (*c1* > *c2*). For the second group, the cognitive and social components are using the same value of acceleration factor (*c1* = *c2*). While for the third group, the value of acceleration coefficient for the cognitive component is less than the value of acceleration coefficient for social component (*c1* < *c2*). Initially, all these three sub-groups have the same population size within their group for example in Figure 3-2, all sub-groups consists of ten particles.

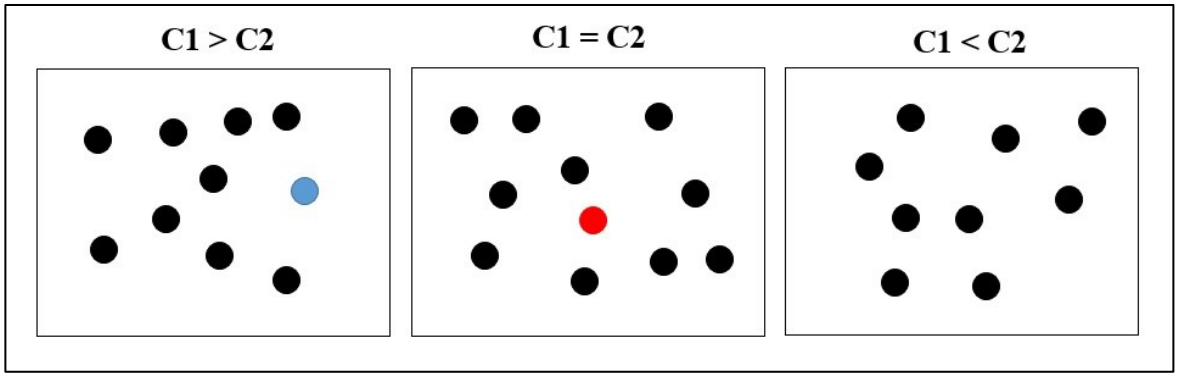


Figure 3-4. Dynamic Acceleration Coefficients PSO (DACPSO) after five iterations.

This proposed method is also using the reward and punishment approach where after five iterations, the group that produces the fittest solutions in these five iterations is awarded an additional populations taken from the two losing groups. The acceleration coefficients used for the losing groups also will be reduced by 10% in comparison to the winning group. For example in the Figure 3-3, after five iterations, the third group is the best performing group which will be awarded with 1 particle (10% of total population) from the first and second groups.

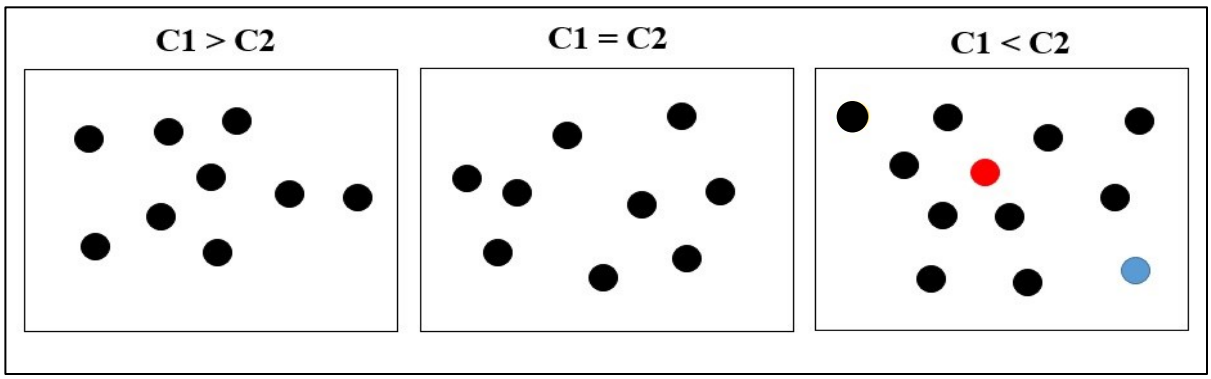


Figure 3-5. Dynamic Acceleration Coefficients PSO (DACPSO) the awarding and punishing process.

Figure 3-4 illustrates the condition of the population size of each group after the awarding and punishing process applied. The winning group now consists of 20% extra population compared to the losing groups as illustrated in Figure 3-5. However, this state is not final as this process is repeating for every five iterations until the maximum iteration numbers are reached or the global optimal is found.

3.3.3 Constricted Area Extended Particle Swarm Optimisation (CAEPSO)

Area Extended PSO (AEPSO) was introduced several years ago and has been used in several optimisation problems including machine learning and navigation problems (A. Atyabi & Phon-Amnuaisuk, 2007; A. Atyabi & Powers, 2013; Adham Atyabi, Phon-Amnuaisuk, & Ho, 2010). This algorithm uses the similar approach as DPPSO where the sub-swarm produces several options of fitness, which are derived from the combination of PSO components. As a result, the fittest sub-swarm is selected to represent the swarm. The idea of modification of this AEPSO are based on several researches which showed Constricted PSO (CPSO) can outperform Linear Decreasing Inertia Weight PSO (Linear PSO) (Bai, 2010; Parsopoulos & Vrahatis, 2011; Syed Abdullah, Hussin, Harun, & Abd Khalid, 2012; Trelea, 2003; H. Zhu et al., 2013). Since AEPSO is utilising Linear PSO as its foundation, the CAEPSO is compared against the original AEPSO. The list of sub-swarm for CAEPSO are as follow:

- Sub-swarm 1: Velocity with Weight

$$V_i(t+1) = K(wV_i(t))$$

- Sub-swarm 2: Cognitive

$$V_i(t+1) = K(c_1 r(P_{best} - x_i(t)))$$

- Sub-swarm 3: Social

$$\mathrm{V_i(t+1)} = K(\mathrm{c_2 r(G_{best} - x_i(t))})$$

- Sub-swarm 4: Velocity with Weight + Cognitive

$$\mathrm{V_i(t+1)} = K(\mathrm{wV_i(t) + c_1 r(P_{best} - x_i(t))})$$

- Sub-swarm 5: Velocity with Weight + Social

$$V_i(t+1) = K(wV_i(t) + c_2 r(G_{best} - x_i(t)))$$

- Sub-swarm 6: Cognitive + Social

$$V_i(t+1) = K(c_1 r(P_{best} - x_i(t)) + c_2 r(G_{best} - x_i(t)))$$

- Sub-swarm 7: Velocity with Weight + Cognitive + Social (Basic PSO)

$$V_i(t+1) = K(wV_i(t) + c_1 r(P_{best} - x_i(t)) + c_2 r(G_{best} - x_i(t)))$$

The constricted value, *K* used is 0.7299 with the inertia weight (*w*) value of 0.7299 as well. The value of acceleration coefficients for social and cognitive components are 0.5 and 2.5 respectively. The random value used is within 0 to 1 with linear distribution is applied.

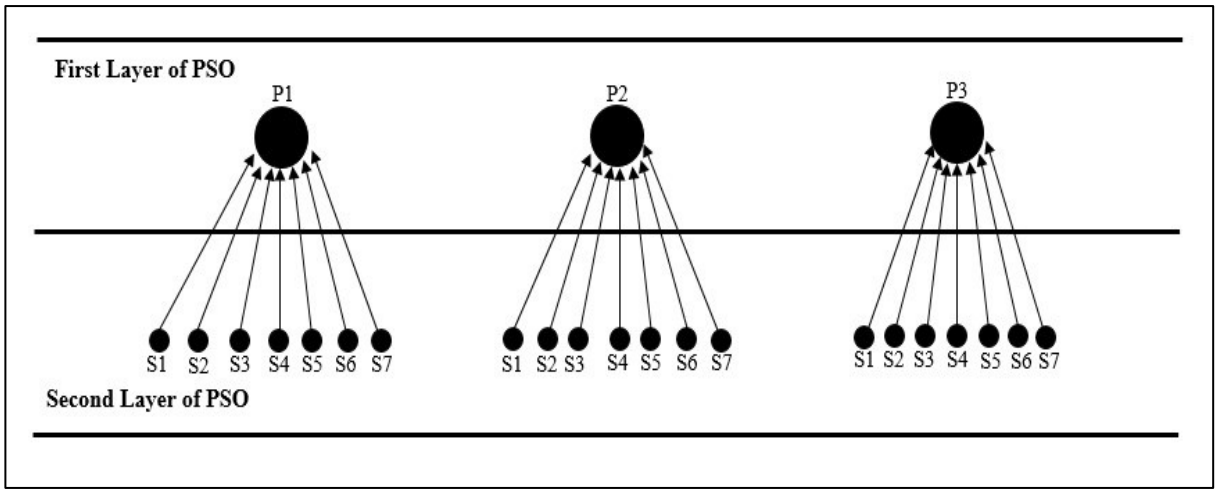


Figure 3-6. Constricted Area Extended PSO (CAEPSO) Architecture.

3.4 Parameter Settings

Parameter settings are crucial for any technique especially swarm intelligence technique. It can determine the outcome of optimal results. Table 3-1 provides a summary of parameter settings utilised in the approaches employed within all experiments. In experiment 1 and 2, benchmark function evaluation and engineering design problem, the population size of all EA approaches are set to 100 with maximum iteration number of 100. It should be noted that all evolutionary approaches examined in other two experiments (maze and symmetric layouts) are set to have a maximum iteration number of 1000 and a population size of 100. All path planning experiments (maze and symmetric layout) are repeated ten times for each method, and the average of the results achieved within each iteration are recorded. All parameter settings applied to the selected algorithm are based on the previous research (Ab Wahab, Nefti-Meziani, & Atyabi, 2015).

Table 3-1. Parameter Setting for Each Algorithm involved.

Algorithm	Parameter Settings
PF	Laser Scan Minimum range set to 0.8.
DA	Several unoccupied points between the starting point and end point are defined before running the algorithm, and full layout of the environment is presented to the algorithm.
RRT	Randomness point is set to 0.5.
PRM	100 random points are created for the next moving point.
GA	Population size is set to 100 with 10% of the population considered as best chromosomes in selection stage. Mutation rate is set 0.05% where mutation operation is to be selected if fitness is not improved in 5 consecutive iterations.
DE	Population size is set to 100 with crossover constant being set to 0.5.

CSA	p_a is set to 0.25, and a maximum number of nests are set to 100.
Fix PSO	Inertia weight is set to 0.7 (Fix Inertia Weight). Cognitive and social coefficients are set to 0.5 and 2.5 respectively (Fix Acceleration Coefficients).
Rand PSO	Random values between 1.0 and 0.5 are used for inertia weight (RANDIW). Cognitive and social coefficients are set to random values number between 2.5 and 0.5 (RANDAC).
TVAC PSO	Linear decreasing value is used for all inertia weight (LDIW) and Time-Varying for Acceleration Coefficients (TVAC). Inertia weight is set to 0.9 as the starting value and 0.4 as the end value. For the social component, the acceleration coefficient is set to 2.5 as the starting value and decreased to 0.5 with the respect of iteration. For the cognitive component, the acceleration coefficient value is set to 0.5 initially and increased to the value of 2.5 towards the end of the iterations.
Linear PSO	Linear decreasing approach varying from 0.9 to 0.4 is utilised for linear decreasing inertia weight (LDIW). The cognitive and social coefficient is set to the fixed values of 0.5 and 2.5 respectively (FAC).
CPSO	The cognitive and social coefficient is set to fixed values of 0.5 and 2.5 respectively (FAC) with constricted value, K of 0.7299.
MPSO	The same parameter settings used in Linear PSO with Inertia Weight (IW) decreasing from 0.9 to 0.4 over iterations with fixed values for acceleration coefficients. The morphology coefficients for social, a_g and cognitive, a_p are set to 0.005.
CMPSO	The parameter settings for this algorithm are exactly as MPSO with an introduction of constricted constant, K with the value of 0.7299.
AEPSO	Linear PSO configuration is taken as the reference for AEPSO parameter settings where the inertia weight (IW) varies from 0.9 to 0.4 and fixed accelerations for cognitive and social with 0.5 and 2.5 respectively.
CAEPSO	Constricted PSO configuration is considered for this algorithm with constricted value, K of 0.7299, identical to the one used in CPSO.

DACPSO	The linear decreasing approach is used for inertia weight with three groups of sub-swarm with different acceleration coefficients. The first sub-swarm used the value of 2.5 for cognitive and 0.5 for social. The second sub-swarm used equal coefficient value, which is 1.5. The last sub-swarm used the value of 0.5 and 2.5 for cognitive and social component respectively.
DPPSO	The behaviour of this particular method is covered in the previous chapter. As a recap, this method combines four commonly used parameter settings in PSO, which are Fix PSO, Rand PSO, TVAC PSO and Linear PSO.

3.5 Summary

This chapter has explained in details all five proposed methods and also parameters setup that will be used in the experiments designed. These five methods introduced are believed to be able to help overcome two major problems in swarm intelligence discussed earlier, which are suitability to implement on the online application and the challenge in fine tuning or finding appropriate parameter settings for swarm intelligence. These two problems can give a researcher a difficult time especially in configuring the parameter setup. In order to assess the performance of these proposed methods, four different experiments have been designed. The following chapter discussed the performance of selected existed evolutionary algorithms against the proposed methods on thirteen benchmark functions.

CHAPTER 4

THE BENCHMARK FUNCTIONS EXPERIMENT

4.1 Introduction

There are many optimisation algorithms claiming superiority over other techniques. Therefore, to decide the most reliable algorithms, benchmark functions can be utilised as a gauge to prove their efficiency. Several benchmark functions with different characteristics have been utilised to measure the ability of the discussed optimisation algorithms; their achieved performances are presented in this section. The first experiment is the comparison between seven algorithms discussed with more severe conditions to determine the best basic evolutionary algorithm.

4.2 Benchmark Functions

In this experiment, the performance of optimisation techniques selected are assessed on a variety of benchmark functions using MATLAB2011 on a CORE i7 CPU with 2GB RAM and was run a hundred times. Table 4-1 presents the list of benchmark functions utilised to assess the performance of the alleged evolutionary methods. The table contains the name of the benchmark function, the characteristic of the function, the dimension, the range, and its equation. The features of the function determine its complexity.

Table 4-1. List of Benchmark Functions involved in the Experiment.

Function	Formula	Value	Dim	Range	Properties
Beale	$f(x)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$	0	2	[-4.5, 4.5]	Unimodal, Inseparable
Bohachevsky1	$f(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1)-0.4cos(4\pi x_2)+0.7$	0	2	[-100, 100]	Multimodal, Separable
Bohachevsky2	$f(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1)cos(4\pi x_2)+0.3$	0	2	[-100, 100]	Multimodal, Inseparable
Bohachevsky3	$f(x)=x_1^2+2x_2^2-0.3cos(3\pi x_1+4\pi x_2)+0.3$	0	2	[-100, 100]	Multimodal, Inseparable
Booth	$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	0	2	[-10, 10]	Multimodal, Separable
Branin	$f(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1^2-6)^2+10(1-\frac{1}{8\pi})\cos x_1+10$	0.3979	2	[-5, 10] x [0, 15]	Multimodal, Separable
Easom	$f(x)=-cos(x_1)cos(x_2)$ $\exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right)$	0	30	[-30, 30]	Unimodal, Inseparable
GoldStein-Price	$f(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right]$ $\times\left[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right]$	3	2	[-10, 10]	Multimodal, Inseparable
Hump	$f(x)=4-2.1\frac{x_1^4}{3})x_1^2+x_1x_2+)-4$ $+4x_2^2)x_2^2$	0	2	[-3, 3] x [-2, 2]	Multimodal, Inseparable
Matyas	$f(x)=0.26(x_1^2+x_2^2)-0.48x_1x_2$	0	2	[-10, 10]	Unimodal, Inseparable
Rastrigin	$f(x)=\sum_{i=1}^{n}x_i^2-10\cos(2\pi x_i)+10$	0	30	[-5.12, 5.12]	Multimodal, Separable
Sphere	$f(x)=\sum_{i=1}^{n}x_i^2$	0	30	[-100, 100]	Unimodal, Separable
Sumsquare	$f(x)=\sum_{i=1}^{n}ix_i^2$	0	30	[-5.12, 5.12]	Unimodal, Separable

Each benchmark function has a combination of unimodal or multimodal with separable or non-separable to make its properties. The combination of these characteristics defines the difficulty of the benchmark functions. A function with two or more local optima is considered as multimodal, and it is defined separable if it can be rewritten as an addition of a function just

from one variable. The theory of epistasis or interrelation between variables of the function is linked to separable properties. The theory of epistasis is a model of genetics where the result of one genetic factor can be governed by the presence of one or more altered genetic factor. The problem can become more complex if the function is multimodal. The value of global optimum is the desired information during the search process. Hence, the regions around local minima must be circumvented. The local optima which spread randomly in the search area are considered as the most difficult problem. As the main aim of optimisation process is to achieve the global optima, therefore the regions around local optima should be circumvented to prevent the swarm get stuck in local optima value and considering the local optima value as the global optima value. Another significant property which determines the complexity of the optimisation problem is the dimension of the search area.

4.3 Result for Morphology Particle Swarm Optimisation

This benchmark function experiment consists of thirteen functions with different types of properties. The results recorded the average result of the runs (Mean), standard deviation (SD) and time taken (in seconds) to complete each hundred iterations. All results are reported in Table 4-2 and Table 4-3. If the mean value is less than 1.000e-12, then the result is reported as 0.000e+00. The first benchmark function was Beale function with unimodal and inseparable properties and a theoretical minimisation value of zero. None of any algorithms managed to find this optimal value. However, Constricted PSO (CPSO) had become the best performing algorithm with an average result of 5.290e-03 which was the closest to optimal value compared to others. The second best performing algorithm Beale function was Time-Varying Acceleration Coefficients PSO (TVAC PSO) with 6.726e-03 average outcome. In the

Bohachevsky1 function, none of the algorithms achieved the best minimisation performance once again but this time Cuckoo Search Algorithm (CSA) had become the best algorithm with 4.224e-07 and Morphology PSO (MPSO) had become the second best with 5.110e-04. The third best was Fixed PSO (Fix PSO) where it managed to achieve 7.411e-04. The results in Bohachevsky2 function indicated that Differential Evolution (DE), Genetic Algorithm (GA) and MPSO have accomplished the optimal value of 0.0 and trailed by Constricted Morphology PSO (CMPSO) with 6.809e-07. DE, GA, MPSO and CMPSO achieved better minimisation performance compared to the other approaches when applied to the Bohachevsky3 functions (with zero being the optimal value once again). CSA became the next best performer with 4.773e-07 on average.

Random PSO (Rand PSO) and TVAC PSO managed to achieve optimal value and became the best performing method on the Booth function followed by MPSO with the mean value of 1.076e-11. Linear PSO (Linear PSO) managed to outperform other approaches by achieving 3.979e-01 mean value which was the closest to the theoretical optimum value of 0.398 on the Branin function. In Easom function, Rand PSO, Linear PSO and CPSO including proposed algorithms (MPSO and CMPSO) were considered as the best performing methods with all of them achieved the mean value of -1.000e+00. DE almost achieved the optimal value with an average result of -9.831e-01. More than half algorithms managed to achieve the theoretical optimal value which was 3.000e+00 with the Goldstein-Price function except for four algorithms (DE, GA, CSA and Fix PSO). MPSO managed to outperform other approaches by achieving 3.952e-08 mean value which was the closest to the theoretical optimum value of zero on the Hump function. The four final functions were Matyas, Rastrigin, Sphere and Sumsquare function. Rand PSO and TVAC PSO managed to be the best performing approaches

(achieved optimum value which was zero) in three functions except for Rastrigin function where MPSO managed to outperform them with an average of 2.894e-02. MPSO also managed to achieve optimum in Sumsquare function.

The results indicated the superiority of MPSO over other methods where it outperformed other techniques in seven out of thirteen benchmark functions. This performance was followed closely by Rand PSO by becoming the best performing method in six of the benchmark functions and being the second best performing approach on a few benchmark functions. The third best performing method was TVAC PSO where it outclassed other algorithms in five functions. Linear PSO, CPSO and CMPSO shared the same number of best performing algorithm of three. The least performing algorithm was Fix PSO where it failed to outperform others or achieved the optimal value in any benchmark functions. It is noteworthy that, concerning the average time spent to finish the optimisation problem, the proposed PSO (MPSO and CMPSO) performed considerably faster (e.g., approximately between 2 to 60 times faster) than the other algorithms.

The results presented in Table 4-6 and Table 4-7 can also be investigated based on the characteristics of the fitness functions utilised in the research (summarised in Table 4-4). Considering the characteristics of i) Unimodal (U), ii) Multimodal (M), iii) Separable (S), iv) Inseparable (I), Unimodal and Separable (US), v) Unimodal and Inseparable (UI), vi) Multimodal and Separable (MS), vii), and viii) Multimodal and Inseparable (MI). Unimodal benchmark function consists of five functions (Beale, Easom, Matyas, Sphere and Sumsquare) where Rand PSO managed to find global optima value and became the best performing algorithm in four functions. The Second best performing algorithm in this category was TVAC PSO with three best performance out of five. Other functions than those five functions were

considered in Multimodal category functions (Bohachevsky1, Bohachevsky2, Bohachevsky3, Booth, Branin, Goldstein-Price, Hump and Rastrigin). In this category, MPSO was the superior approach with five times as the best performing algorithm. Other algorithms (except CSA, Fix PSO, and CPSO) were tied as the second best performing algorithms with two best performers in total. Separable functions were Bohachevsky1, Booth, Branin, Rastrigin, Sphere and Sumsquare. Surprisingly, half of the algorithms failed to become performing approaches even once. Those algorithms were DE, GA, Fix PSO, CPSO and CMPSO. The best performing algorithms for this category were Rand PSO and TVAC PSO with three times as the best performing algorithms. MPSO closely followed it with two times. Beale, Bohachevsky2, Bohachevsky3, Easom, Goldstein-Price, Hump, and Matyas functions fell under inseparable category. MPSO managed to outperform other algorithms five times and became overall best performing algorithm in this category. The second best algorithm was shared by Rand PSO, CPSO and MPSO with three times as the best performing method. CSA and Fix PSO, however, struggled to become the best performing approach in any benchmark function.

The following categories were the combination of two types of properties. The first type was Unimodal and Separable, which consisted of Sphere and Sumsquare functions. Only Rand PSO and TVAC managed to become the best performing algorithm in both functions while MPSO managed to become the best performing algorithm once in Sumsquare function. However, other function failed to achieve the optimal value at all. The second category was Unimodal and Inseparable, and the best performing approach was shared between Rand PSO and CPSO where they managed to beat other algorithm two out of three times. The Second best performing algorithm was shared among TVAC PSO, Linear PSO, MPSO and CMPSO where they achieved optimal value once. The next combination was Multimodal and Separable where

four functions (Bohachevsky1, Booth, Branin and Rastrigin) fell under this category. None of the algorithms managed to come out top with five algorithms only once succeeded to become the best performing method. Those five algorithms were CSA, Rand PSO, TVAC PSO, Linear PSO and MPSO. Multimodal and Inseparable combination was the final category considered with four functions fell under this category as well. The functions were Bohachevsky2, Bohachevsky3, Goldstein-Price and Hump function. MPSO showed its superiority in this category where it outperformed others in four occasions. The second best approach with two best performances was DE, GA, and CMPSO. Only CSA and Fix PSO failed to get the best performance in any function under this category.

Considering the results presented and the discussion, MPSO seems to be the best overall performing approach, outperforming other methods in seven out of thirteen functions followed by Rand PSO with the best performance in six out of thirteen. The third best is TVAC PSO with five out of thirteen best performance. Linear PSO, CPSO, and CMPSO reached the best performance in three out of thirteen functions. With an attention on the breakdown results, it is obvious that MPSO has been the best performing method in four out of eight categories. Nonetheless, in terms of execution time to finish the benchmark tests, Fix PSO, MPSO, and CMPSO are the best with an average of fewer than 0.1 seconds for all functions. Although Rand PSO and TVAC are the best second and third overall performance in term of mean value, it is the least fast algorithm.

Table 4-2. MPSO Results for Benchmark Optimisation Problems

Benchmark Function	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	MPSO	CMPSO
Beale Optimum(0)	1.844e-02 1.055e-01 3.5707s	1.438e-01 1.920e-01 2.4946s	9.680e-01 5.312e-01 0.2230s	7.419e-03 5.352e-02 0.0840s	3.697e-02 2.002e-01 9.8281s	6.726e-03 5.329e-02 0.0768s	2.898e-02 1.039e-01 0.0782s	**5.290e-03** **4.881e-01** **0.1683s**	6.961e-02 4.123e-01 0.0727s	8.207e-03 6.231e-02 0.0665s
Bohachecsky1 Optimum (0)	9.365e-02 4.194e-03 3.9298s	5.932e-02 5.961e-03 3.9723s	**4.224e-07** **5.192e-07** **0.2273s**	7.411e-04 4.594e-03 0.0977s	7.900e-03 5.686e-04 0.1011s	9.161e-04 5.034e-03 0.0749s	1.194e-03 6.099e-03 0.0790s	4.133e-03 3.287e-01 0.1658s	5.110e-04 6.974e-03 0.0819s	8.326e-04 7.189e-03 0.0797s
Bohachecsky2 Optimum (0)	**0.000e+00** **0.000e+00** **3.7869s**	**0.000e+00** **0.000e+00** **2.3782s**	9.931e-07 1.450e-06 0.2313s	7.140e-04 3.174e-03 0.1019s	2.097e-03 1.236e-02 0.1081s	6.289e-04 2.827e-03 0.0796s	5.679e-04 2.844e-03 0.0780s	2.647e-03 1.491e-01 0.1672s	**0.000e+00** **0.000e+00** **0.0753s**	6.809e-07 6.314e-06 0.0741s
Bohachecsky3 Optimum (0)	**0.000e+00** **0.000e+00** **3.8314s**	**0.000e+00** **0.000e+00** **2.3734s**	4.773e-07 7.198e-07 0.2222s	1.861e-03 1.326e-02 0.1125s	3.246e-03 2.551e-02 0.1075s	1.911e-03 1.333e-02 0.0819s	2.061e-03 1.391e-02 0.0827s	1.086e-02 1.016e+00 0.1699s	**0.000e+00** **0.000e+00** **0.0814s**	**0.000e+00** **0.000e+00** 0.0713s
Booth Optimum(0)	4.619e-04 4.188e-02 0.1083s	1.762e-02 1.241e+00 0.1069s	4.625e-02 4.188e-01 0.0782s	1.651e-01 6.782e-01 0.0800s	**0.000e+00** **0.000e+00** **3.8136s**	**0.000e+00** **0.000e+00** **2.3698s**	2.189e-09 2.332e-09 0.2203s	2.426e-10 2.227e-09 0.1701s	1.076e-11 1.321e-10 0.0735s	9.958e-08 5.165e-07 0.0735s
Branin Optimum (0.398)	5.041e-01 9.006e-01 0.1109s	5.265e-01 1.286e+00 0.1114s	4.986e-01 8.999e-01 0.0772s	8.368e-01 1.724e+00 0.0814s	4.210e-01 2.308e-01 3.6172s	4.534e-01 3.586e-02 2.6820s	**3.979e-01** **1.046e-09** **0.3192s**	4.143e-01 9.819e+00 0.1719s	4.135e-01 7.156e-01 0.0817s	4.633e-01 5.321e-01 0.0727s
Easom Optimum (0)	-9.831e-01 1.204e-01 0.1119s	-9.786e-01 1.356e-01 0.1131s	-9.830e-01 1.204e-01 0.0794s	-7.989e-05 6.379e-06 0.0808s	**-1.000e+00** **0.000e+00** **3.5815s**	-9.534e-01 1.525e-02 2.3623s	**-1.000e+00** **0.000e+00** **0.2262s**	**-1.000e+00** **0.000e+00** **0.1702s**	**-1.000e+00** **0.000e+00** **0.0783s**	**-1.000e+00** **0.000e+00** **0.0674s**

Table 4-3. MPSO Results for Benchmark Optimisation Problems (cont'd.)

Benchmark Function	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	MPSO	CMPSO
Goldstein-Price Optimum(3)	3.019e+00 1.105e-01 0.1153s	3.212e+00 1.594e+00 0.1000s	3.019e+00 1.103e-01 0.0920s	3.032e+00 1.700e-01 0.0939s	**3.000e+00** **0.000e+00** **3.5528s**	**3.000e+00** **0.000e+00** **2.4216s**	**3.000e+00** **0.000e+00** **0.2214s**	**3.000e+00** **0.000e+00** **0.1861s**	**3.000e+00** **0.000e+00** **0.0704s**	**3.000e+00** **0.000e+00** **0.0883s**
Hump Optimum(0)	2.254e-03 1.338e-02 0.12137s	4.771e-04 3.023e-03 0.1253s	2.698e-03 1.485e-02 0.0896s	2.312e-03 1.546e-02 0.0880s	4.651e-08 7.744e-17 3.5223s	5.033e-01 3.994e-01 2.4592s	5.125e-08 5.137e-09 0.2382s	7.742e-04 8.169e-02 0.1782s	**3.952e-08** **3.465e-07** **0.0827s**	1.569e-07 3.164e-06 0.0766s
Matyas Optimum(0)	3.168e-05 2.523e-04 0.1167s	6.203e-05 5.600e-04 0.1092s	2.868e-05 2.515e-04 0.0816s	3.600e-05 2.715e-04 0.0827s	**0.000e+00** **0.000e+00** **3.7907s**	**0.000e+00** **0.000e+00** **2.3762s**	9.220e-10 1.488e-09 0.2251s	2.610e-06 5.287e-02 0.1672s	3.168e-05 5.312e-04 0.0861s	3.253e-05 7.321e-04 0.0711s
Rastrigin Optimum(0)	5.382e-02 2.881e-01 0.1164s	3.794e-02 2.163e-01 0.0978s	4.219e-02 2.726e-01 0.0833s	5.345e-02 2.810e-01 0.081433s	-1.985e+01 4.093e-01 3.7988s	-1.786e+01 3.652e+00 2.8458s	-1.169e+01 1.238e-04 0.2186s	8.940e-02 3.865e+00 0.1722s	**2.894e-02** **6.311e-01** **0.0714s**	1.061e+00 3.135e+00 0.0707s
Sphere Optimum(0)	1.836e-04 1.312e-03 0.0867s	1.792e-04 1.295e-03 0.0791s	1.773e-04 1.306e-03 0.0757s	4.154e-04 2.129e-03 0.0796s	**0.000e+00** **0.000e+00** **3.8280s**	**0.000e+00** **0.000e+00** **2.4001s**	1.073e-10 1.718e-10 0.2011s	6.431e-05 5.241e-02 0.1680s	1.814e-04 2.311e-03 0.0619s	9.569e-05 3.134e-04 0.0758s
Sumsquare Optimum (0)	6.257e-05 4.003e-04 0.0661s	4.199e-04 3.366e-03 0.0613s	5.903e-05 3.936e-04 0.0665s	1.054e-04 5.466e-04 0.0630s	**0.000e+00** **0.000e+00** **3.8473s**	**0.000e+00** **0.000e+00** **2.3893s**	2.107e-10 2.946e-10 0.2066s	9.382e-05 5.221e-02 0.1524s	**0.000e+00** **0.000e+00** **0.0523s**	7.672e-05 7.232e-04 0.0537s
Best Performing Algorithm	2	2	1	0	6	5	3	3	7	3

Table 4-4. MPSO Results for Benchmark Optimisation Problems based on Benchmark Category.

Category	Number of functions	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	MPSO	CMPSO
Unimodal (U)	5	0	0	0	0	4	3	1	2	2	1
Multimodal (M)	8	2	2	1	0	2	2	2	1	5	2
Separable (S)	6	0	0	1	0	3	3	1	0	2	0
Inseparable (I)	7	2	2	0	0	3	2	2	3	5	3
Unimodal Separable (US)	2	0	0	0	0	2	2	0	0	1	0
Unimodal Inseparable (UI)	3	0	0	0	0	2	1	1	2	1	1
Multimodal Separable (MS)	4	0	0	1	0	1	1	1	0	1	0
Multimodal Inseparable (MI)	4	2	2	0	0	1	1	1	1	4	2
Being best performing method	13	2	2	1	0	6	5	3	3	7	3

4.4 Significance Analysis for Morphology Particle Swarm Optimisation

Table 4-5. Significant Analysis for Benchmark Optimisation Problems (MPSO).

Category	Benchmark Functions	Evolutionary Methods	Benchmark Functions & Evolutionary Methods
Fitness Value	$p = 0$	$p = 0$	$p = 0$
Time	$p = 0$	$p = 0$	$p = 0$

With the purpose of measuring the significance of the performance achieved and to offer a fair valuation of these achievements, statistical significance analysis is applied to the results using N-Way ANOVA and Kruskal-Wallis tests. Lilliefors test is used before determining the parametric nature of the results. The statistical significance is assessed on both benchmark functions and evolutionary approaches. For Benchmark Functions Experiments, only two factors are considered and analysed for the statistical difference.

The first category of analysing for significance is the fitness or the outcome of the function. The statistical analysis of the results (Table 4-5) indicates the existence of statistical significance between the performance achieved from various benchmark functions ($p = 0 < 0.5$), different evolutionary methods ($p = 0 < 0.5$) and the interactions of the benchmark functions and evolutionary methods ($p = 0 < 0.05$).

Among the benchmark functions, Beale, Bohachevsky1, Bohachevsky2, Bohachevsky3, Easom and Rastrigin show a significantly different from each other and all other benchmark functions. Booth, Branin, Goldstein-Price, Hump, Matyas, Sphere and SumSquare indicate a lack of statistical significance amongst each other, but they are significantly different from Beale, Bohachescky1, Easom, and Rastrigin functions.

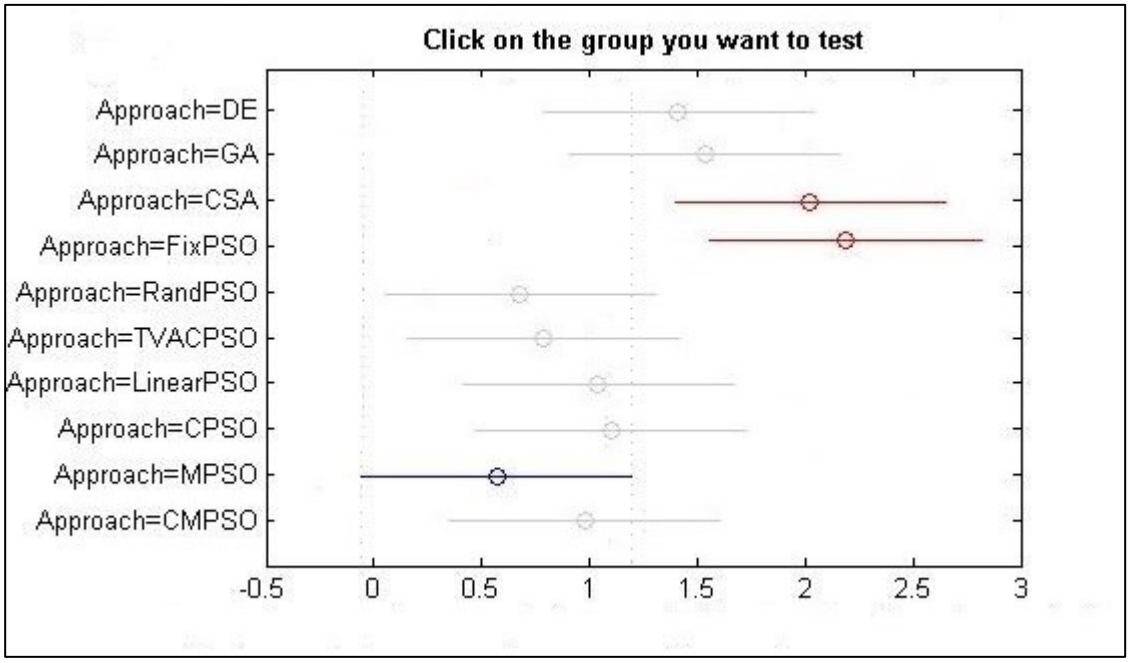


Figure 4-1. Box Plot of Significant Difference in Benchmark Optimisation Problems for Mean Error factor.

Between the evolutionary methods (Figure 4-1), Rand PSO, CPSO, and MPSO are showing no significant differences with each other while they are significantly different from CSA and Fix PSO. TVAC PSO is showing statistical significance against Fix PSO but not against the other algorithms involved.

The second category considered is Execution Time (*s*). The statistical analysis of the results indicates significant difference between the computation time of benchmark functions ($p = 0 < 0.05$), evolutionary methods ($p = 0 < 0.05$) and the interactions of the evolutionary methods and the benchmark functions ($p = 0 < 0.05$).

The results indicate that among benchmark functions, no significant difference is being observed between the results in Beale, Bohachescky2, Bohachescky3, Booth, Goldstein-Price, Hump, Matyas, Sphere and SumSquare. However, Beale is significantly different from Bohachescky1, Branin and Rastrigin functions. Bohachescky1 is significantly distinct from all other benchmark functions. Bohachescky2, Bohachescky3, Booth, Matyas, Sphere and SumSquare are only significantly different from Bohachescky1 and Rastrigin functions. Branin and Easom are significantly different from Bohachescky1, Beale, Goldstein-Price and Hump functions. Goldstein-Price and Hump are only showing significant differences from Bohachescky 1, Branin, Easom and Rastrigin functions.

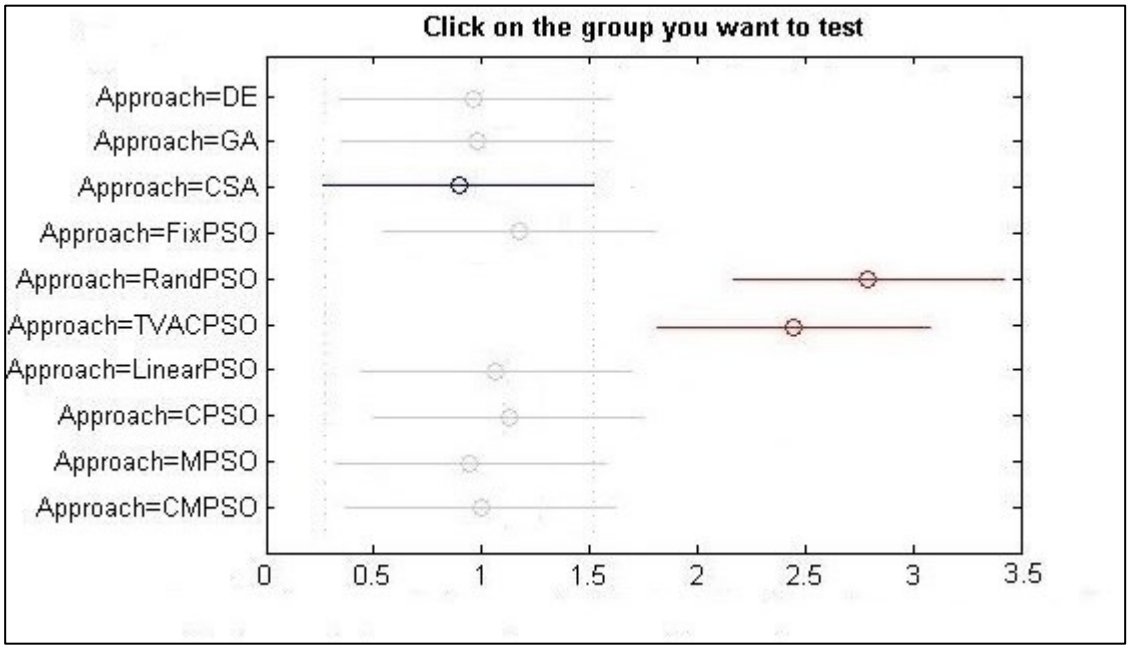


Figure 4-2. Box Plot of Significant Difference in Benchmark Optimisation Problems for Execution Time factor.

Between the evolutionary methods (Figure 4-2), most of the variations of PSO (e.g., Fix PSO, Linear PSO, CPSO, MPSO, and CMPSO) are lacking significant differences from each

other while they all show such significance when they are compared with Rand PSO and TVAC PSO. Rand PSO and TVAC PSO indicate lack significant differences from each other.

4.5 Result for Dynamic Approaches of Particle Swarm Optimisation

The similar benchmark functions are used for assessment of Dynamic Approaches of Particle Swarm Optimisation as well which consists of thirteen functions with different types of properties. The results recorded are the average result of the runs (Mean), standard deviation (SD) and time taken (in seconds) to complete each hundred iterations. All results achieved are reported in Table 4-6 and Table 4-7. The same approach for results reported is used, if the mean value is less than 1.000e-12, then the result is reported as 0.000e+00. Constricted Area Extended PSO (CAEPSO) and Dynamic Parameterising PSO (DPPSO) were the only algorithms managed to achieve global optima value (0.000e+00) for Beale function. The second best performing algorithm was Area Extended PSO (AEPSO) with an average output of 2.555e-10. The next function was Bohachevsky1 with an optimal value of zero. In this function, three algorithms (AEPSO, CAEPSO, and DPPSO) managed to achieve the optimal value. The second best performing algorithm was CSA with an average final output of 4.224e-07. Bohachevsky2 had the same optimum value as the previous function which was zero and this time the same three algorithms (AEPSO, CAEPSO, and DPPSO) managed to find the optimum value once again together with DE and GA. The second best algorithm was CSA once again with an average of 9.931e-07.

DE, GA, AEPSO, CAEPSO, and DPPSO successfully found the optimal value for Bohachevsky3 and became the best performing algorithm. Fix PSO became the second best performing algorithm with an overall mean value of 1.861e-03. Another function that had zero

optimum value was Booth where Rand PSO, TVAC PSO, AEPSO, CAEPSO, Dynamic Acceleration Coefficients PSO (DACPSO), and DPPSO managed to achieve it. CPSO almost achieved the global optima value as well but slightly short with 2.426e-10. AEPSO, CAEPSO, DACPSO and DPPSO shared the same average output which was 3.980e-01 and was the optimal value for Branin function. Linear PSO almost achieved the optimal value too with an mean outcome of 3.979e-01. Seven algorithms successfully found the optimal value for Easom function which was -1. Those seven methods were Rand PSO, Linear PSO, CPSO, AEPSO, CAEPSO, DACPSO, and DPPSO. DE recorded average output of -9.831e-01 and made it as the second best performing algorithm. With an optimum value of 3, Goldstein-Price function had the most successful algorithm to achieve those value. Eight out of twelve algorithms achieved it except DE, GA, CSA and Fix PSO. DE and CSA did come close to optimal value with an average of 3.019e-00. None of the algorithms managed to find the optimal value for Hump function. However, DPPSO managed to outperform other algorithms with 3.610e-09 and CAEPSO came second with 6.463e-09. Rand POS, TVAC PSO, AEPSO, CAEPSO, and DPPSO were the best performing algorithm for Matyas Function with an average output of optimal value of zero. The second best performing algorithm was Linear PSO with a mean output of 9.220e-10.

Only two algorithms (CAEPSO and DPPSO) managed to achieve optimum value and became the best performing algorithm for Rastrigin function. The second best was GA with an average output of 3.794e-02. For Sphere and Sumsquare functions, both functions had an optimal value of zero. The same algorithms managed to achieve the optimal value for both functions. The algorithms were Rand PSO, TVAC PSO, AEPSO, CAEPSO, DACPSO, and DPPSO. Hence, based on the observation and discussion, DPPSO becomes the overall best

performing algorithm where it outperforms other methods in thirteen occasions. CAEPSO is just slightly short with twelve out of thirteen best performing algorithm and becomes the second best. The third best algorithm belongs to AEPSO with outstanding performance on ten occasions. The fourth place is shared by Rand PSO and DACPSO where both of them show excellent performances in seven out of thirteen benchmark functions. In terms of time, Fix PSO and CSA dominate in almost all functions utilised where they outperform others quite well. However, they are short regarding the main objective which is minimising or maximising the benchmark function across hundred runs. Therefore, it is proven that fast convergence does not guarantee the best outcomes.

From an observation on Table 4-6 and Table 4-7, the following discussion and analysis are based on the properties or characteristic of the function (summarised in Table 4-8). The same considering characteristics used which are i) Unimodal (U), ii) Multimodal (M), iii) Separable (S), iv) Inseparable (I), Unimodal and Separable (US), v) Unimodal and Inseparable (UI), vi) Multimodal and Separable (MS), vii), and viii) Multimodal and Inseparable (MI). Beale, Easom, Matyas, Sphere, and Sumsquare function fall under Unimodal category where CAEPSO and DPPSO share the top spot as the best algorithm by outperforming other methods in five benchmark functions. Rand PSO and AEPSO come second with outstanding performances in four benchmark functions. Meanwhile, the third best spot is shared between TVAC, PSO, and DACPSO with decent performances in three benchmark functions. The Next category is Multimodal where the rest of unknown functions in Unimodal are under this category. The top performance for this category is DPPSO with eight best performances and followed by CAEPSO and AEPSO with seven and six outstanding performances respectively. Bohachevsky1, Booth, Branin, Rastrigin, Sphere, and Sumsquare are considered in Separable

category. DE, GA, CSA, Fix PSO, Linear PSO and CPSO fail to achieve best performing algorithm even once under this category. The best performing algorithm under this category is DPPSO and CAEPSO with six out of six functions. AEPSO comes as second best once again with five out of six functions. The Third best performing algorithm is DACPSO with four out of six benchmark functions. DPPSO manages to become the best performing algorithm with seven outstanding performances under the Inseparable category that consists seven functions in it. The next four categories are a combination of two properties of Unimodal and Multimodal with Separable and Inseparable. The first combination is between Unimodal and Separable where two functions (Sphere and Sumsquare) are involved, and six algorithms manage to achieve the optimal value of those functions. Those six algorithms are Rand PSO, TVAC PSO, AEPSO, CAEPSO, DACPSO, and DPPSO.

The second combination is Unimodal and Inseparable which consists of three functions (Beale, Easom and Matyas). CAEPSO and DPPSO manage to outperform others in all three functions. The second best performing algorithm belongs to Rand PSO and AEPSO with outstanding performances in two out of three functions. The third best performing spot is shared between TVAC PSO, Linear PSO, CPSO and DACPSO with top performance in one function. Bohacehvsky1, Booth, Branin and Rastrigin are four functions considered under Multimodal and Separable category. CAEPSO and DPPSO come out on top once again with outstanding performances in all functions considered. AEPSO and DACPSO come out second with excellent performances in three out of four occasions. The third best performance under this category is Rand PSO with two decent performances.

The final category considered is Multimodal and Inseparable and four functions (Bohachevsky2, Bohachevsky3, Goldstein-Price and Hump) are listed under this group.

DPPSO is the best performing algorithm where it manages to outperform others in all functions. The second best performing algorithm is joint by AEPSO and CAEPSO with three best performance out of four. DE and GA share the third spot with two top performances. Considering the results presented and analysed, DPPSO is considered as the best overall performing approach, outperforming other approaches in all thirteen functions followed closely by CAEPSO with the outstanding performance in twelve out of thirteen. The third best is AEPSO with ten out of thirteen best performance. Rand PSO and DACPSO have reached the best performance in seven out of thirteen functions, and that makes them as the fourth best algorithm. By concentrating on the breakdown results, it is obvious that DPPSO has been the most outstanding performing method in eight out of eight categories and CAEPSO is left as second but with only a slight shortage.

Table 4-6. DAPSO Results for Benchmark Optimisation Problems.

Benchmark Function	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	AEPSO	CAEPSO	DACPSO	DPPSO
Beale	1.844e-02	1.438e-01	9.681e-01	7.420e-03	3.697e-02	6.726e-03	2.898e-02	5.290e-03	2.555e-10	**0.000e+00**	5.6132e-03	**0.000e+00**
Optimum(0)	1.055e-01	1.921e-01	5.306e-01	5.353e-02	2.002e-01	5.330e-02	1.039e-01	4.882e-01	2.555e-09	**0.000e+00**	5.6130e-02	**0.000e+00**
	3.5707s	2.4946s	0.2230s	0.0840s	9.8281s	0.0768s	0.0782s	0.1683s	0.5346s	**0.4621s**	0.0984s	**0.5132s**
Bohachecsky1	9.365e-02	5.932e-02	4.224e-07	7.411e-04	7.900e-03	9.161e-04	1.194e-03	4.133e-03	**0.000e+00**	**0.000e+00**	9.3407e-02	**0.000e+00**
Optimum (0)	4.194e-03	5.961e-03	5.192e-07	4.594e-03	5.686e-04	5.034e-03	6.099e-03	3.287e-01	**0.000e+00**	**0.000e+00**	1.8784e-01	**0.000e+00**
	3.9298s	3.9723s	0.2273s	0.0977s	0.1011s	0.0749s	0.0790s	0.1658s	**0.5463s**	**0.4975s**	0.9012s	**0.5013s**
Bohachecsky2	**0.000e+00**	**0.000e+00**	9.931e-07	7.140e-04	2.097e-03	6.289e-04	5.679e-04	2.647e-03	**0.000e+00**	**0.000e+00**	1.9648e-02	**0.000e+00**
Optimum (0)	**0.000e+00**	**0.000e+00**	1.450e-06	3.174e-03	1.236e-02	2.827e-03	2.844e-03	1.491e-01	**0.000e+00**	**0.000e+00**	6.2792e-02	**0.000e+00**
	3.7869s	2.3782s	0.2313s	0.1019s	0.1081s	0.0796s	0.0780s	0.1672s	**0.5823s**	**0.4963s**	0.09832s	**0.5277s**
Bohachecsky3	**0.000e+00**	**0.000e+00**	4.773e-07	1.861e-03	3.246e-03	1.911e-03	2.061e-03	1.086e-02	**0.000e+00**	**0.000e+00**	1.1313e-02	**0.000e+00**
Optimum (0)	**0.000e+00**	**0.000e+00**	7.198e-07	1.326e-02	2.551e-02	1.333e-02	1.391e-02	1.016e-00	**0.000e+00**	**0.000e+00**	4.9561e-02	**0.000e+00**
	3.8314s	**2.3734s**	0.2222s	0.1125s	0.1075s	0.0819s	0.0827s	0.1699s	**0.5941s**	**0.4712s**	0.1321s	**0.4912s**
Booth	4.619e-04	1.762e-02	4.625e-02	1.651e-01	**0.000e+00**	**0.000e+00**	2.189e-09	2.426e-10	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**
Optimum(0)	4.188e-02	1.241e-00	4.188e-01	6.782e-01	**0.000e+00**	**0.000e+00**	2.332e-09	2.227e-09	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**
	0.1083s	0.1069s	0.0782s	0.0800s	**3.8136s**	**2.3698s**	0.2203s	0.1701s	**0.7012s**	**0.6178s**	**0.1078s**	**0.6012s**
Branin	5.041e-01	5.265e-01	4.986e-01	8.368e-01	4.210e-01	4.534e-01	3.979e-01	4.143e-01	**3.980e-01**	**3.980e-01**	**3.980e-01**	**3.980e-01**
Optimum	9.006e-01	1.286e-00	8.999e-01	1.724e-00	2.308e-01	3.586e-02	1.046e-09	9.819e-00	**9.8623e-04**	**4.6543e-04**	**2.3892e-08**	**1.7075e-02**
(0.398)	0.1109s	0.1114s	0.0772s	0.0814s	3.6172s	2.6820s	0.3192s	0.1719s	**0.7221s**	**0.6245s**	**0.1489s**	**0.6132s**
Easom	-9.831e-01	-9.786e-01	-9.830e-01	-7.989e-05	**-1.000e+00**	-9.534e-01	**-1.000e+00**	**-1.000e+00**	**-1.000e+00**	**-1.000e+00**	**-1.000e+00**	**-1.000e+00**
Optimum(-1)	1.204e-01	1.356e-01	1.204e-01	6.379e-06	**0.000e-00**	1.525e-02	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**	**0.000e+00**
	0.1119s	0.1131s	0.0794s	0.0808s	**3.5815s**	2.3623s	**0.2262s**	**0.1702s**	**0.7333s**	**0.6423s**	**0.1678s**	**0.6519s**

Table 4-7. DAPSO Results for Benchmark Optimisation Problems (cont'd.).

Benchmark Function	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	AEPSO	CAEPSO	DACPSO	DPPSO
Goldstein-Price Optimum(3)	3.019e-00 1.105e-01 0.1153s	3.212e-00 1.594e-00 0.1000s	3.019e-00 1.103e-01 0.0920s	3.032e-00 1.700e-01 0.0939s	**3.000e-00** **1.326e-15** **3.5528s**	**3.000e-00** **0.000e-00** **2.4216s**	**3.000e-00** **2.782e-06** **0.2214s**	**3.000e-00** **0.000e-00** **0.1861s**	**3.000e-00** **0.000e-00** **0.4841s**	**3.000e-00** **0.000e-00** **0.4111s**	**3.000e-00** **0.000e-00** **0.1482s**	**3.000e-00** **0.000e-00** **0.4651s**
Hump Optimum(0)	2.254e-03 1.338e-02 0.12137s	4.771e-04 3.023e-03 0.1253s	2.698e-03 1.485e-02 0.0896s	2.312e-03 1.546e-02 0.0880s	4.651e-08 7.744e-17 3.5223s	5.033e-01 3.994e-01 2.4592s	5.125e-08 5.137e-09 0.2382s	7.742e-04 8.169e-02 0.1782s	1.8552e-04 1.8548e-03 0.4765s	6.463e-09 1.1245e-08 0.4019s	4.6510e-08 1.6726e-07 0.1465s	**3.610e-09** **3.2998e-10** **0.4056s**
Matyas Optimum(0)	3.168e-05 2.523e-04 0.1167s	6.203e-05 5.600e-04 0.1092s	2.868e-05 2.515e-04 0.0816s	3.600e-05 2.715e-04 0.0827s	**0.000e+00** **0.000e+00** **3.7907s**	**0.000e+00** **0.000e+00** **2.3762s**	9.220e-10 1.488e-09 0.2251s	2.610e-06 5.287e-02 0.1672s	**0.000e+00** **0.000e+00** **0.4439s**	**0.000e+00** **0.000e+00** **0.3872s**	8.0045e-05 3.4675e-04 0.1237s	**0.000e+00** **0.000e+00** **0.3526s**
Rastrigin Optimum(0)	5.382e-02 2.881e-01 0.1164s	3.794e-02 2.163e-01 0.0978s	4.219e-02 2.726e-01 0.0833s	5.345e-02 2.810e-01 0.081433s	-1.985e+01 4.093e-01 3.7988s	-1.786e+01 3.652e-00 2.8458s	-1.169e+01 1.238e-04 0.2186s	8.940e-02 3.865e-00 0.1722s	7.1710e-02 4.1249e-01 0.4664s	**0.000e+00** **0.000e+00** **0.3773s**	1.154e+00 4.1760e-01 0.1256s	**0.000e+00** **0.000e+00** **0.3651s**
Sphere Optimum(0)	1.836e-04 1.312e-03 0.0867s	1.792e-04 1.295e-03 0.0791s	1.773e-04 1.306e-03 0.0757s	4.154e-04 2.129e-03 0.0796s	**0.000e+00** **0.000e+00** **3.8280s**	**0.000e+00** **0.000e+00** **2.4001s**	1.073e-10 1.718e-10 0.2011s	6.431e-05 5.241e-02 0.1680s	**0.000e+00** **0.000e+00** **0.4510s**	**0.000e+00** **0.000e+00** **0.3615s**	**0.000e+00** **0.000e+00** **0.1456s**	**0.000e+00** **0.000e+00** **0.3213s**
Sumsquare Optimum (0)	6.257e-05 4.0029e-04 0.0661s	4.199e-04 3.366e-03 0.0613s	5.903e-05 3.936e-04 0.0665s	1.054e-04 5.466e-04 0.0630s	**0.000e+00** **0.000e+00** **3.8473s**	**0.000e+00** **0.000e+00** **2.3893s**	2.107e-10 2.946e-10 0.2066s	9.382e-05 5.221e-02 0.1524s	**0.000e+00** **0.000e+00** **0.4101s**	**0.000e+00** **0.000e+00** **0.3336s**	**0.000e+00** **0.000e+00** **0.1659s**	**0.000e+00** **0.000e+00** **0.3013s**
Best Performing Algorithm	1	1	0	0	7	5	2	2	10	12	7	13

Table 4-8. DAPSO Results for Benchmark Optimisation Problems based on Benchmark Category.

Category	Number of functions	DE	GA	CSA	Fix PSO	Rand PSO	TVAC PSO	Linear PSO	CPSO	AEPSO	CAEPSO	DACPSO	DPPSO
Unimodal (U)	5	0	0	0	0	4	3	1	2	4	5	4	5
Multimodal (M)	8	2	2	1	0	2	2	2	1	6	7	3	8
Separable (S)	6	0	0	1	0	3	3	1	0	5	6	4	6
Inseparable (I)	7	2	2	0	0	3	2	2	3	5	6	3	7
Unimodal Separable (US)	2	0	0	0	0	2	2	0	0	2	2	2	2
Unimodal Inseparable (UI)	3	0	0	0	0	2	1	1	2	2	3	2	3
Multimodal Separable (MS)	4	0	0	1	0	1	1	1	0	3	4	2	4
Multimodal Inseparable (MI)	4	2	2	0	0	1	1	1	1	3	3	1	4
Being best performing method	13	2	2	1	0	6	5	3	3	10	12	7	13

4.6 Significance Analysis for Dynamic Approach of Particle Swarm Optimisation

Table 4-9. Significance Analysis for Benchmark Optimisation Problems (DAPSO).

Category	Benchmark Functions	Evolutionary Methods	Benchmark Functions & Evolutionary Methods
Fitness Value	$p = 0$	$p = 0$	$p = 0$
Time	$p = 0$	$p = 0$	$p = 0$

In order to calculate the significance of the performance achieved and to offer a fair valuation of these achievements, the same statistical significance analysis from the previous section is applied to the results by using N-Way ANOVA and Kruskal-Wallis tests with Lilliefors test that is used earlier to decide the parametric characteristic of the results. The statistical significance is assessed on both benchmark functions and evolutionary methods. For Benchmark Functions Experiments, the same two factors are considered and analysed for statistical differences.

The fitness or the outcome of the function is considered as the first category analysed for significance. The statistical analysis of the results (refer Table 4-9) points out the existence of statistical significance between the performance achieved from various benchmark functions ($p = 0 < 0.5$) and various evolutionary methods ($p = 0 < 0.5$) including the interactions of the benchmark functions and evolutionary methods ($p = 0 < 0.05$).

Between the benchmark functions, Beale, Bohachevsky1, Bohachevsky2, Branin, Hump, and Rastrigin have indicated significant differences from all other benchmark functions but not amongst themselves. Bohachevsky3, Booth, Easom, Goldstein-Price, Matyas, Sphere, and SumSquare functions have indicated lack of statistical significance amongst each other, but

they are significantly different form Beale, Bohachevsky1, Bohachevsky2, Branin, Hump, and Rastrigin functions.

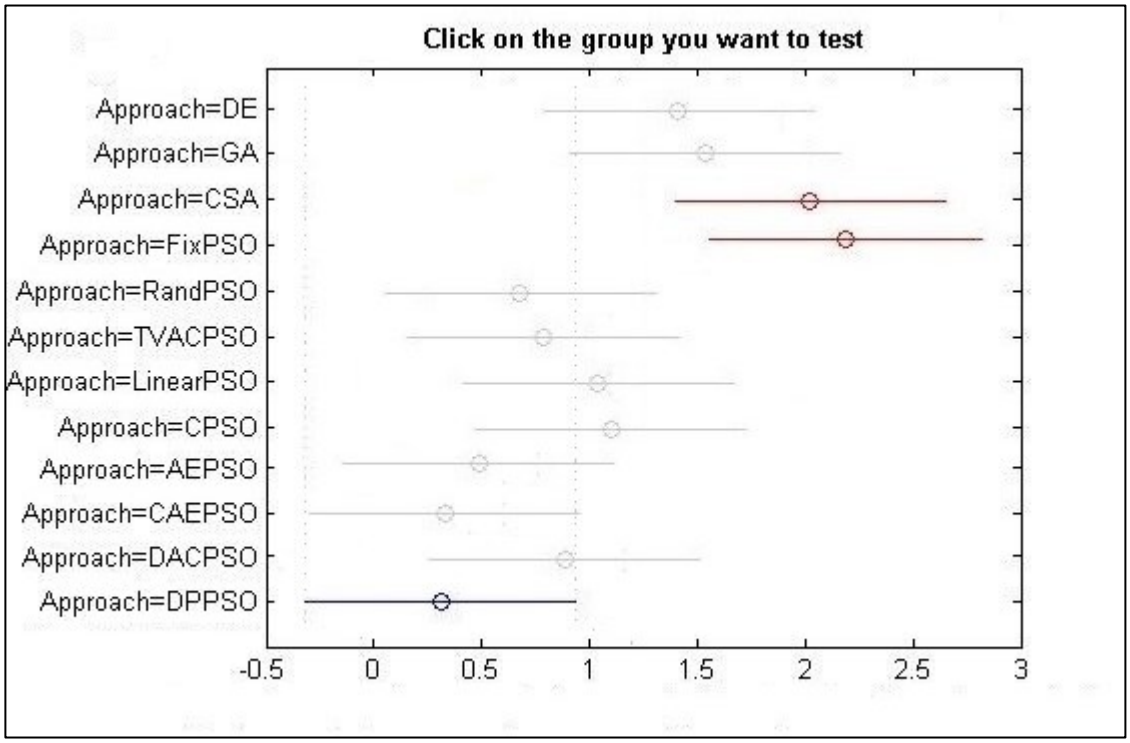


Figure 4-3. Box Plot of Significant Difference in Benchmark Optimisation Problems for Mean Error factor.

Between the evolutionary methods (Figure 4-3), only Rand PSO, AEPSO, CAEPSO, and DPPSO have indicated significant differences against CSA and Fix PSO but a lack of significant difference against each other. Meanwhile, the other algorithms are shown a lack of significant different between themselves.

The Execution Time (s) is considered as the second category. The statistical analysis of the results points out significant differences between the computation time of benchmark functions (p=0<0.05), evolutionary methods (p=0<0.05) and the interactions of the evolutionary methods and the benchmark functions (p=0<0.05).

The results indicate that among benchmark functions, there is a significant difference observed between the Beale, Bohachescky3, Booth, Branin, Matyas, Rastrigin, Sphere and SumSquare. Easom, Goldstein-Price and Hump have shown lack of significant differences amongst each other. However, these three functions have shown significant differences against Bohachescky1 and Bohachescky2. Bohachescky1 and Bohachescky2 are significantly different from all other benchmark functions.

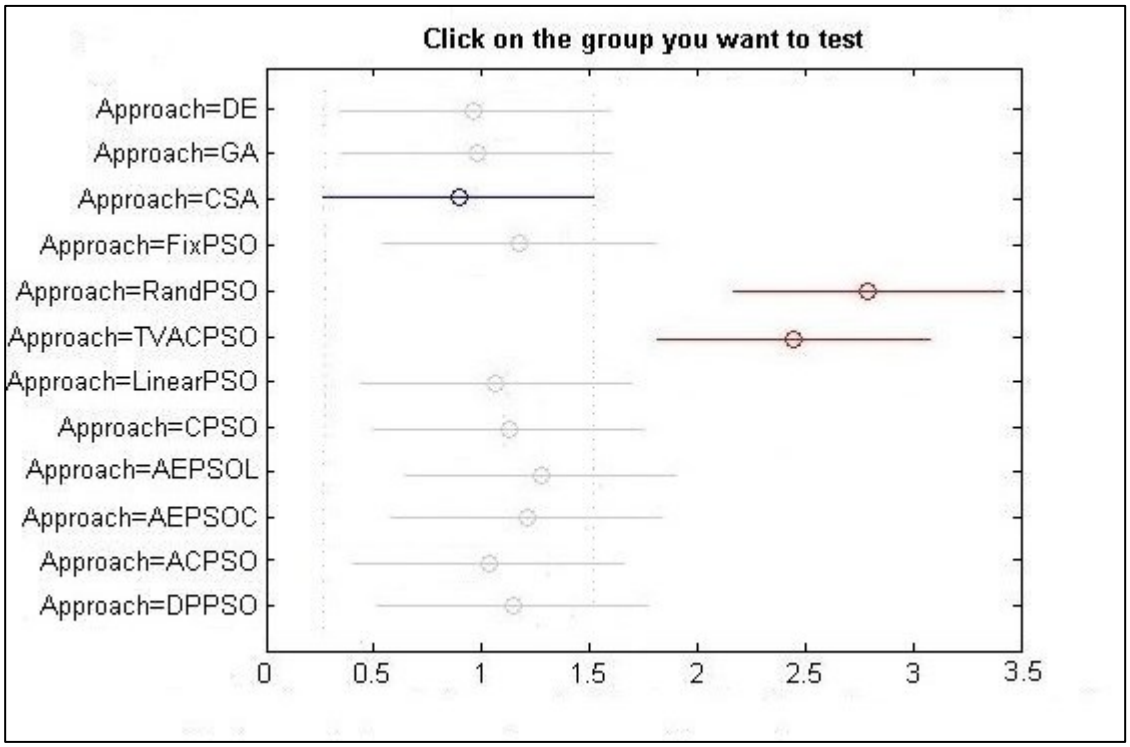


Figure 4-4. Box Plot of Significant Difference in Benchmark Optimisation Problems for Execution Time factor.

For the evolutionary algorithms category (Figure 4-4), the proposed PSOs (AEPSO, CAEPSO, DACPSO and DPPSO) lack significant difference from each other including against CSA, Fix PSO, Linear PSO and CPSO while they all show such significance when they are compared with Rand PSO and TVAC PSO. Meanwhile, Rand PSO and TVAC PSO have shown lack significant differences from each other.

4.7 Summary

This section discusses about the performance of proposed algorithms against existing evolutionary approaches in thirteen benchmark functions. Based on the results and discussion from this benchmark functions experiments, the proposed algorithms have shown a promising performance against other evolutionary approaches including the variant of PSOs. CAEPSO and DPPSO are the most outstanding performing algorithms with excellent performances across all benchmark functions implemented. DACPSO also shows the decent performance against existing evolutionary algorithm. It matches Rand PSO as the best performing algorithm in seven benchmark functions. MPSO shows a promising performance too as it is selected as the best performing algorithm under the comparison for morphology PSO against other evolutionary methods. CMPSO shows a glimpse of excellent performance in several benchmark functions but slightly short compared to MPSO. The next section will be focusing on more complex optimisation problems with several constraints. The following optimisation problems consist of three well-known engineering design problems.

CHAPTER 5

ENGINEERING DESIGN PROBLEMS

5.1 Introduction

Engineering Design Problems is one of the common practise used to evaluate the performances of any swarm intelligence techniques. The complexity and precision are needed to find the optimal value of the output with several limitations that need to be considered. In Chapter 4, three well-known engineering design problems involved in this chapter have been discussed in details. Each engineering design problems results and discussion for Morphology PSO and Dynamic Approaches of PSO are presented including the significance analysis studies on them.

5.2 Types of Engineering Design Problems

For Engineering Design Problem Experiments, three well-known and common design problems among researchers are selected. These are the Spring Design Optimisation Problem, the Welded Beam Design Problem and the Pressure Vessel Design Problem. The details for each Engineering Design Problem are discussed in the following subsection. The code has been implemented on the same platform as the previous experiment which is MATLAB2011 on a CORE i7 CPU with 2GB RAM and has been run a hundred times as well.

5.2.1 Tension/Compression Design Optimisation Problem

Tensional or compressional springs are commonly used in engineering. A general spring design problem consists of three design variables (refer Figure 5-1); the wire diameter *w*, the mean coil diameter *d*, and the length (or number of coils) *L*. The objective for this design problem is to minimise the weight of the spring with several constraints such as maximum shear stress, minimum deflection, and geometrical limits.

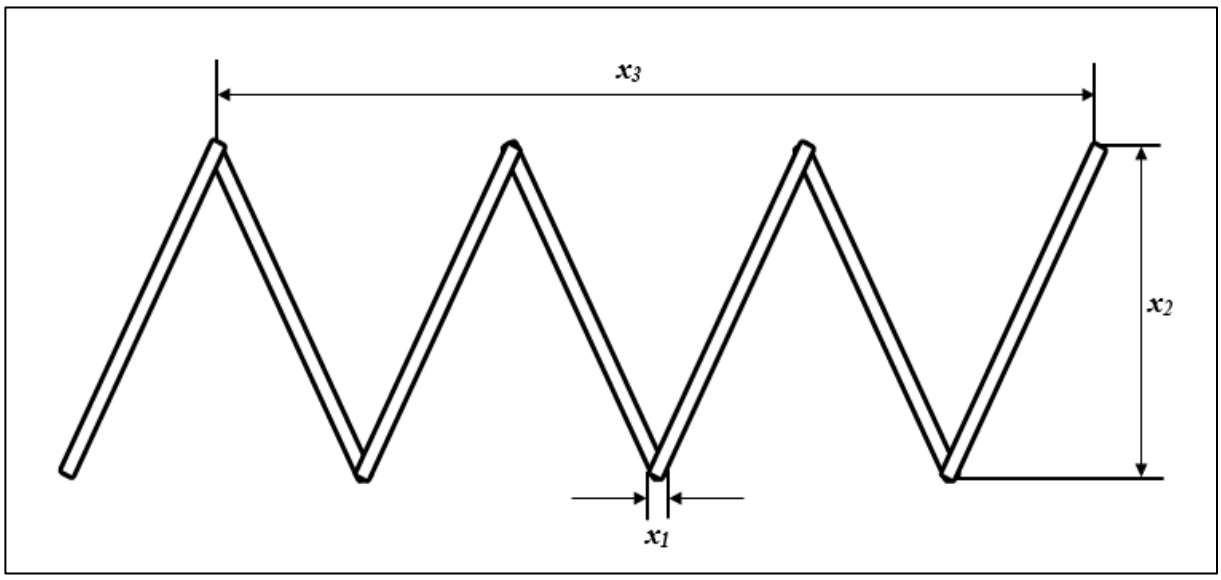


Figure 5-1. Design of the Tension/Compression String Problem.

The detailed description can be referred to earlier studies (Cagnina, Esquivel, & Coello Coello, 2008). This problem can be written compactly as:

Minimise $f(x) = (L+2)w^2 d$ *Equation 25*

subject to

$g_1(x) = 1 - \frac{d^3 L}{71785 w^4} \leq 0,$ *Equation 26*

$$g_2(x) = 1 - \frac{140.45w}{d^2 L} \leq 0, \qquad \textit{Equation 27}$$

$$g_3(x) = \frac{2(w+d)}{3} - 1 \leq 0, \qquad \textit{Equation 28}$$

$$g_4(x) = \frac{d(4d-w)}{w^4(12566d-w)} + \frac{1}{5108w^2} - 1 \leq 0, \qquad \textit{Equation 29}$$

with the following limits

$$0.05 \leq w \leq 2.0, 0.25 \leq d \leq 1.3, 2.0 \leq L \leq 15.0$$

5.2.2 Welded Beam Design Optimisation Problem

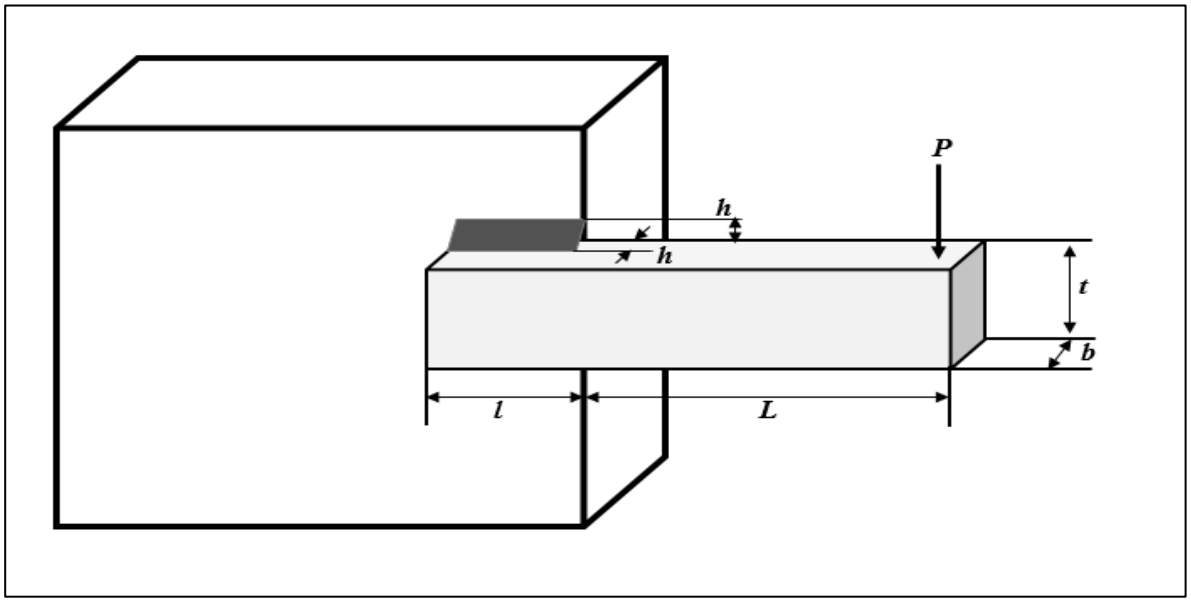


Figure 5-2. Design of Welded Beam Problem.

Another regular assessment problem for constrained design optimisation is the welded beam design as illustrated in Figure 5-2 (Cagnina et al., 2008). This design problem consists of four design variables: length *L* and the width *w* of the welded area, thickness *h* and the depth *h*

of the main beam. The aim of this problem is to minimise the overall fabrication cost with the constraints of bending stress σ, buckling load P, shear stress τ, and maximum end deflection δ. The problem can be written as minimise:

$$f(x) = 1.10471w^2L + 0.04811dh(14.0 + L), \quad \textit{Equation 30}$$

subject to

$$g_1(x) = w - h \leq 0, \quad \textit{Equation 31}$$

$$g_2(x) = \delta(x) - 0.25 \leq 0, \quad \textit{Equation 32}$$

$$g_3(x) = \tau(x) - 13600 \leq 0, \quad \textit{Equation 33}$$

$$g_4(x) = \sigma(x) - 30000 \leq 0, \quad \textit{Equation 34}$$

$$g_5(x) = 0.10471w^2 + 0.04811hd(14 + L) - 5.0 \leq 0, \quad \textit{Equation 35}$$

$$g_6(x) = 0.125 - w \leq 0, \quad \textit{Equation 36}$$

$$g_7(x) = 6000 - P(x) \leq 0, \quad \textit{Equation 37}$$

where

$$\sigma(x) = \frac{504000}{hd^2}, \quad \textit{Equation 38}$$

$$Q = 6000(14 + \frac{L}{2}) \quad \textit{Equation 39}$$

$$D = \frac{1}{2}\sqrt{L^2 + (w + d)^2}, \quad \textit{Equation 40}$$

$$J = \sqrt{2}wL\left[\frac{L^2}{6} + \frac{(w+d)^2}{2}\right], \quad \textit{Equation 41}$$

$$\delta = \frac{65856}{30000hd^3}, \quad \textit{Equation 42}$$

$$\beta = \frac{QD}{J}, \quad \textit{Equation 43}$$

$$\alpha = \frac{6000}{\sqrt{2}wL}, \quad \textit{Equation 44}$$

$$\tau(x) = \sqrt{\alpha^2 + \frac{\alpha\beta L}{D} + \beta^2}, \quad \textit{Equation 45}$$

$$P = 0.61423 \times 10^6 \frac{dh^3}{6}\left(1 - \frac{\sqrt[d]{30/48}}{28}\right) \quad \textit{Equation 46}$$

The boundaries or limits are $0.1 \leq L$, $h \leq 2.0$, $d \leq 10$, and $0.1 \leq w$.

5.2.3 Pressure Vessel Design Optimisation Problem

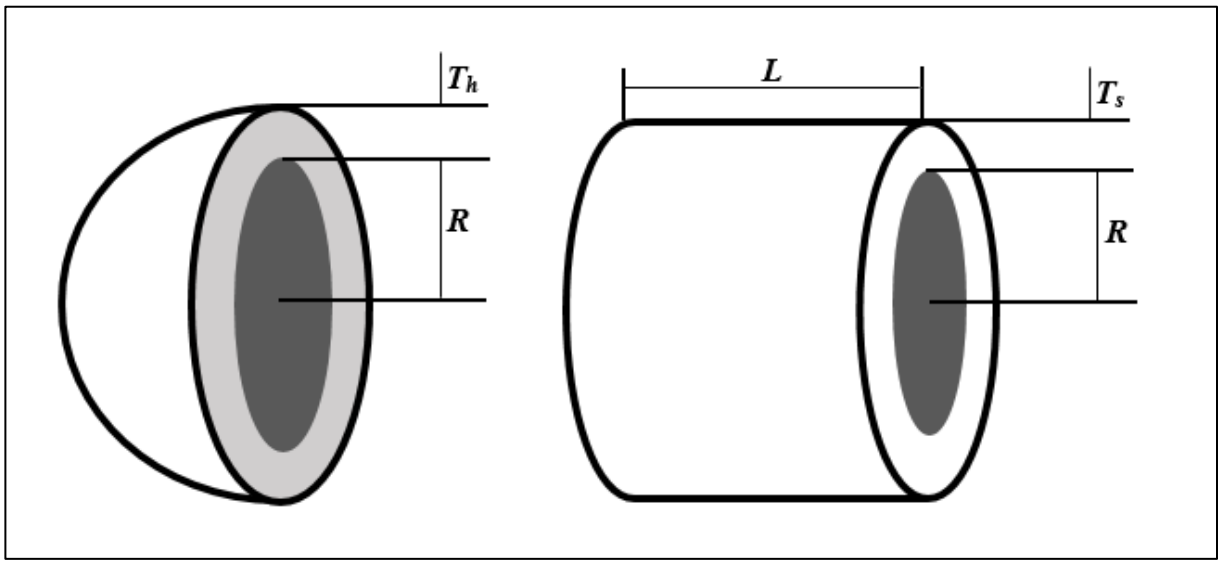


Figure 5-3. Pressure Vessel Design Problem.

Figure 5-3 illustrated a cylindrical pressure vessel capped at both ends by hemispherical heads. This compressed air storage tank has a working pressure of 3000 psi with a maximum volume of 750 ft^3. It is designed based on the ASME boiler and pressure vessel code. The objective is to minimise total cost, including a combination of single welding cost, material and forming cost (Choi & Chang, 2013). The variables involved are the thickness (T_s), the length of the cylindrical section of the vessel (L), the thickness of the head (T_h), and the inner radius (R). The thicknesses of the variables are discrete values, which are integer multiples of 0.0625 inch. The mathematical modelling for this optimisation problem can be summarised as follows:

Minimize:

$$f(T_s, T_h, R, L) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_h^2 K \qquad \textit{Equation 47}$$

Subject to:

$$g_1 = -T_s + 0.0193R \leq 0 \qquad \textit{Equation 48}$$

$$g_2 = -T_h + 0.0095R \leq 0 \qquad \textit{Equation 49}$$

$$g_3 = -\pi R^2 L - \frac{4}{3}\pi R^3 + 1296000 \leq 0 \qquad \textit{Equation 50}$$

$$g_4 = L - 240 \leq 0 \qquad \textit{Equation 51}$$

where $1 \times 0.0625 \leq T_S, T_h \leq 99 \times 0.0625$, and $10 \leq R, L \leq 200$.

5.3 Result for Morphology Particle Swarm Optimisation

Eight existing evolutionary algorithms in literature are implemented on three Engineering Design Problems Optimisation and compared against MPSO and CMPSO. The average results of hundred executions for Tension/Compression Design Problem, Welded Beam Design Problem, and Pressure Vessel Design Problem are presented in Table 5-1, Table 5-2, and Table 5-3 respectively. For Tension/Compression Design Problem, this problem consists of three variables named *x1*, *x2,* and *x3* which represent the main coil diameter, the wire diameter, and the number of coils respectively. The best performing algorithm is MPSO with the lowest average outcome of 0.01266 with design variables of 0.05175, 0.35818, and 11.20376. Another four algorithms are just slightly short of being the best performing algorithm

with an average outcome of 0.01267. CMPSO obtains the best output for *x1* (the main coil diameter) with an average of 0.05115. Meanwhile, the best outcome for *x2* (the wire diameter) and *x3* (the number of coils) are obtained by GA and TVAC PSO respectively with 0.32158 and 8.68448. Although GA has one of the best outcomes for design variables, but it also has the highest average outcome of *x3* with 13.97994. Hence, its final average outcome is the highest with 0.01310 compared to other methods.

Table 5-1. MPSO results for Tension/Compression Design Problem.

Tension/Compression Design Problem				
Algorithm	Design Variables			*f(x)*
	x1	*x2*	*x3*	
DE	0.05341	0.39922	9.18541	0.01271
GA	0.05046	**0.32158**	13.97994	0.01310
CSA	0.05216	0.36816	10.64844	0.01267
Fix PSO	0.05164	0.35536	11.39793	0.01274
Rand PSO	0.05173	0.35764	11.24454	0.01267
TVAC PSO	0.05395	0.41137	**8.68448**	0.01279
Linear PSO	0.05340	0.39918	9.18540	0.01273
CPSO	0.05169	0.35669	11.29048	0.01267
MPSO	0.05175	0.35818	11.20376	**0.01266**
CMPSO	**0.05115**	0.34987	12.07643	0.01267

For Welded Beam Design Optimisation Problem, the design variables represent the thickness of the weld, *h* (*x1*), length of the welded joint, *l* (*x2*), width of the beam, *t* (*x3*), and thickness of the beam, *b* (*x4*) with a main objective to minimise the overall cost of fabrication, (*f*(*x*)). The results indicate the superiority of MPSO with the lowest outcome value of 1.73119

and design variables of 0.20150, 3.56200, 9.04140, and 0.20570. CMPSO becomes the second best performing algorithm with 1.73121 and the same as design variables outcome for MPSO; except in *x4* where it produced an average of 0.20571. The third best performing algorithm is CPSO with a mean result of 2.38112, and average outputs for *x1*, *x2*, *x3,* and *x4* are 0.2445, 6.21867, 8.29154, and 0.2442 respectively. Fix PSO is the least performing algorithm with an output of 2.44116 on average. MPSO and CMPSO share the lowest outcome for *x1* and *x2* with a mean outcome of 0.20150 and 3.56200 respectively. Linear PSO obtains the lowest output for third design variable *x3* with an average of 8.17897. MPSO once again managed to find the lowest value for design variable *x4* compared to other algorithms with a mean value of 0.20570.

Table 5-2. MPSO Results for Welded Beam Design Problem.

Welded Beam Design Problem					
Algorithm	Design Variables				*f(x)*
	x1	*x2*	*x3*	*x4*	
DE	0.24551	6.19600	8.27301	0.24555	2.38591
GA	0.24552	6.19602	8.27445	0.24553	2.38597
CSA	0.24444	6.21775	8.29164	0.24445	2.38107
Fix PSO	0.24894	6.17357	8.18018	0.25355	2.44116
Rand PSO	0.24897	6.17306	8.17896	0.25337	2.43314
TVAC PSO	0.24448	6.23805	8.28865	0.24465	2.38547
Linear PSO	0.24895	6.17304	**8.17891**	0.25332	2.43318
CPSO	0.24445	6.21867	8.29154	0.24442	2.38112
MPSO	**0.20150**	**3.56200**	9.04140	**0.20570**	**1.73119**
CMPSO	**0.20150**	**3.56200**	9.04140	0.20571	1.73121

Pressure Vessel Design Problem main objective is to find the minimum total cost in fabricating the pressure vessel with four design variables involved. These variables are the

thickness, T_s, the thickness of the head, T_h, the inner radius, R, and the length of the cylindrical section of the vessel, L represents by *x1*, *x2*, *x3,* and *x4* in the results recorded in Table 5-3.

Table 5-3. MPSO Results for Pressure Vessel Design Problem.

Pressure Vessel Design Problem					
Algorithm	Design Variables				*f(x)*
	x1	*x2*	*x3*	*x4*	
DE	**0.81250**	**0.43750**	**42.09127**	176.74650	6061.07770
GA	1.12500	0.62500	58.29100	**43.69000**	7198.04280
CSA	**0.81250**	**0.43750**	40.32390	200.00000	6288.74450
Fix PSO	1.12500	0.62500	58.29000	43.69300	7197.70000
Rand PSO	1.12500	0.62500	47.70000	117.70100	8129.10360
TVAC PSO	1.12500	0.62500	58.27890	43.75490	7198.43300
Linear PSO	0.93750	0.50000	48.32900	112.67900	6410.38110
CPSO	**0.81250**	**0.43750**	42.09809	176.64052	6059.74560
MPSO	**0.81250**	**0.43750**	42.09835	176.63775	**6059.72580**
CMPSO	**0.81250**	**0.43750**	42.09740	176.65405	6059.94600

From the observation in Table 5-3, MPSO managed to outperform other algorithms in term of achieving the main objective with an average output of 6059.72580. The second best performing algorithm is CPSO with a mean of 0.01980 short from MPSO in the final output. CMPSO comes in third as best performing algorithm with a mean output of 6059.94600; 0.22020 short from the best performing algorithm. There are five algorithms share the lowest value for *x1* with an average of 0.81250. They also share the lowest value for *x2* with an average outcome of 0.43750. DE manages to find the lowest *x3* value with an average outcome of 42.09127. GA, Fix PSO, and TVAC PSO are amongst the least performing algorithm group although they managed to find lowest value for design variable *x4* with an average outcome of

43.69000, 43.69300, and 43.75490 respectively. The worst performing algorithm is Rand PSO with a mean final result of 8129.10360.

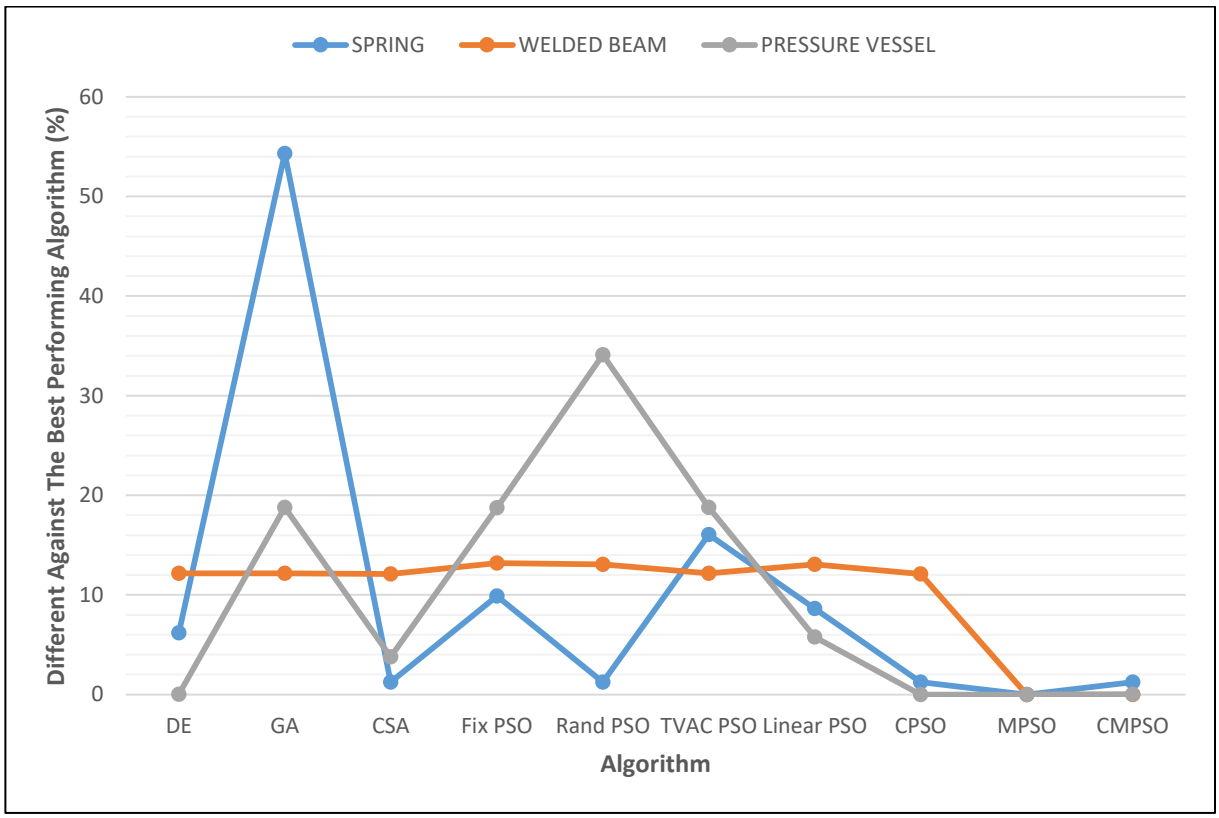


Figure 5-4. Comparison Graph of All Algorithms against the Best Performing Algorithm.

Figure 5-4 illustrates the performance of each method against the best result achieved by the best performing method. MPSO managed to become the best performing algorithm in all three chosen Engineering Design Optimisation Problems (EDP). The second most consistent method is CMPSO where the different against MPSO is less than two percent across all EDP involved. GA recorded the largest different against MPSO in spring design problem with more than half different.

5.4 Significance Analysis for Morphology Particle Swarm Optimisation

The same statistical significance analysis from the previous experiment is applied to the result obtained in this experiment. The statistical tools use is N-Way ANOVA and Kruskal-Wallis with Lilliefors test is applied before defining the parametric nature of the results. The statistical significance is evaluated on benchmark functions and evolutionary approaches. For this Engineering Design Problems, all results from Table 5-1, Table 5-2, and Table 5-3 are considered and analysed for statistical difference and the results are shown in Table 5-4.

Table 5-4. Significance Analysis for Engineering Design Optimisation Problems (MPSO).

Category	Benchmark Functions	Evolutionary Methods	Benchmark Functions & Evolutionary Methods
Fitness Value	$p = 0$	$p = 0$	$p = 0$

The statistical analysis of the results shows the existence of statistical significance between the performance achieved from the design optimisation problems ($p = 0 < 0.5$), the evolutionary algorithms ($p = 0 < 0.5$), and the interactions of the design optimisation problems and evolutionary algorithms ($p = 0 < 0.05$). Welded beam design problem and pressure vessel design problem indicate a lack of significance different between them but show the statistical significance different against tension/compression design problem.

For evolutionary approaches, MPSO and CMPSO show no indication of statistical significance between them including DE, CSA, Linear PSO, and CPSO. However, they demonstrate the occurrence of statistical significance against GA, Fix PSO, Rand PSO, and TVAC PSO as shown in Figure 5-5.

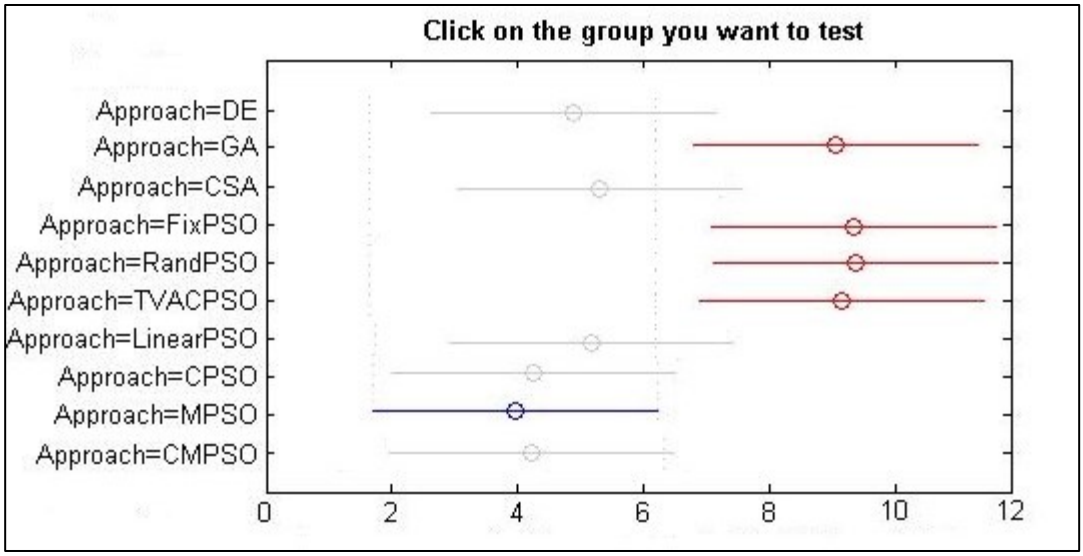


Figure 5-5. Box Plot of Significant Difference for Engineering Design Optimisation Problems (MPSO).

5.5 Result for Dynamic Approaches of Particle Swarm Optimisation

All selected algorithms together with three dynamic approaches PSOs are executed hundred times, and the average results for tension/compression design problem are recorded in Table 5-5. The design variables *x1*, *x2*, and *x3* denote the main coil diameter, the wire diameter, and the number of coils respectively.

Table 5-5. DAPSO Results for Tension/Compression Design Problem.

Tension/Compression Design Problem				
Algorithm	Design Variables			*f(x)*
	x1	*x2*	*x3*	
DE	0.05341	0.39922	9.18541	0.01271
GA	0.05046	**0.32158**	13.97994	0.01310
CSA	0.05216	0.36816	10.64844	0.01267
Fix PSO	0.05164	0.35536	11.39793	0.01274

Rand PSO	0.05173	0.35764	11.24454	0.01267
TVAC PSO	0.05395	0.41137	8.68448	0.01279
Linear PSO	0.05340	0.39918	**9.18540**	0.01273
CPSO	0.05169	0.35669	11.29048	0.01267
AEPSO	0.05175	0.35818	11.20376	0.01266
CAEPSO	0.05168	0.35672	11.28883	**0.01265**
DACPSO	0.05164	0.35536	11.39792	0.01269
DPPSO	**0.05132**	0.35253	11.43886	**0.01265**

DPPSO managed outperforms other algorithms with the lowest average value for *x1* of 0.05132. GA be able to beat other methods with the mean output of 0.32158 for *x2*. Linear PSO recorded 9.18540 average outcomes for *x3* and became the best performing algorithm in this category. However, CAEPSO and DPPSO shared the top spot as the best overall performing approach for this design problem as they recorded the same average final output of 0.01265. Although they share the same outcome but their design variables value are entirely different. CAEPSO recorded 0.05168, 0.35672, and 11.28883 for *x1*, *x2,* and *x3* while DPPSO recorded 0.05132, 0.35253, and 11.43886 for *x1*, *x2,* and *x3*. AEPSO become the second best performing algorithm with only 0.00001 different from CAEPSO and DPPSO with the design variables of 0.05175 (*x1*), 0.35818 (*x2*), and 11.20376 (*x3*).

Table 5-6. DAPSO Results for Welded Beam Design Problem.

Welded Beam Design Problem					
Algorithm	Design Variables				*f(x)*
	x1	*x2*	*x3*	*x4*	
DE	0.24551	6.19600	8.27301	0.24555	2.38591
GA	0.24552	6.19602	8.27445	0.24553	2.38597

CSA	0.24444	6.21775	8.29164	0.24445	2.38107
Fix PSO	0.24894	6.17357	8.18018	0.25355	2.44116
Rand PSO	0.24897	6.17306	8.17896	0.25337	2.43314
TVAC PSO	0.24448	6.23805	8.28865	0.24465	2.38547
Linear PSO	0.24895	6.17304	**8.17891**	0.25332	2.43318
CPSO	0.24445	6.21867	8.29154	0.24442	2.38112
AEPSO	**0.19974**	3.61206	9.03750	0.20608	1.73730
CAEPSO	0.20573	3.46988	9.03671	0.20577	**1.72485**
DACPSO	0.20880	**3.42050**	8.99750	0.21000	1.74831
DPPSO	0.20570	3.47113	9.03668	**0.20573**	1.72492

Four design variables involved in Welded Beam Design Optimisation Problem to minimise the total outcome of the function. Those design variables are the thickness of the weld (*h*), the length of the welded joint (*l*), the width of the beam (*t*), and thickness of the beam (*b*) represent by *x1*, *x2*, *x3,* and *x4* respectively. Twelve algorithms have been executed hundred times to achieve the possible minimal value and compared against each other together with three newly introduced dynamic approaches PSO. The results are shown in Table 5-6. From the observation, CAEPSO manages to achieve the most minimal output compared to other algorithms with an average output of 1.72485 and design variables of 0.20573 (*x1*), 3.46988 (*x2*), 9.03671 (*x3*), 0.20577 (*x4*). DPPSO is the second best performing method with average come of 1.72492 and design variables of 0.20570 (*x1*), 3.47113 (*x2*), 9.03668 (*x3*), 0.20573 (*x4*). Although CAEPSO is the best performing algorithm for this engineering design problem but concerning design variables, CAEPSO did not manage to find the lowest value for any of its design variables. The lowest value for design variables for *x1*, *x2*, *x3,* and *x4* are 0.19974,

3.42050, 8.17891, and 0.20573 accomplished by AEPSO, DACPSO, Linear PSO, and DPPSO respectively.

Table 5-7 illustrates the mean results obtained from running a hundred executions using twelve differences evolutionary algorithms on the pressure vessel design problem. This optimisation problem main objective is to find the minimum cost of designing the pressure vessel with four design variables of the thickness (T_s), the thickness of the head (T_h), the inner radius (R), and the length of the cylindrical section of the vessel (L) which represents by *x1*, *x2*, *x3,* and *x4*. For pressure vessel design problem, DPPSO recorded an average outcome of 6059.71129 and became the best performing algorithm with average design variables of 0.81250 for *x1*, 0.43750 for *x2*, 42.09845 for *x3,* and 176.63655 for *x4*. CAEPSO recorded 6059.71432 on average for final output to become the second best performing algorithm. The average for CAEPSO's design variables are almost similar to DPPSO with the different is only on *x4* with the different of 0.00005. The third best performing algorithm is AEPSO with an average output of 6057.71433 and design variables of 0.81250 (*x1*), 0.43750 (*x2*), 42.09845 (*x3*) and 176.63661 (*x4*). DE, CSA, CPSO, AEPSO, CAEPSO, DACPSO, and DPPSO recorded the same average results for *x1* and *x2* which are 0.81250 and 0.43750 respectively. CSA manages to outperform others with an average of 40.32390 for *x3*. GA recorded 43.6900 on average for *x4* which is the lowest result recorded compared to the other algorithms.

Table 5-7. DAPSO Results for Pressure Vessel Design Problem Design Problem.

Pressure Vessel Design Problem					
Algorithm	Design Variables				*f(x)*
	x1	*x2*	*x3*	*x4*	
DE	**0.81250**	**0.43750**	42.09127	176.74650	6061.07770
GA	1.12500	0.62500	58.29100	**43.69000**	7198.04280

CSA	**0.81250**	**0.43750**	**40.32390**	200.00000	6288.74450
Fix PSO	1.12500	0.62500	58.29000	43.69300	7197.70000
Rand PSO	1.12500	0.62500	47.70000	117.70100	8129.10360
TVACPSO	1.12500	0.62500	58.27890	43.75490	7198.43300
Linear PSO	0.93750	0.50000	48.32900	112.67900	6410.38110
CPSO	**0.81250**	**0.43750**	42.09809	176.64052	6059.74560
AEPSO	**0.81250**	**0.43750**	42.09845	176.63661	6059.71433
CAEPSO	**0.81250**	**0.43750**	42.09845	176.63660	6059.71432
DACPSO	**0.81250**	**0.43750**	42.09809	176.64052	6059.74560
DPPSO	**0.81250**	**0.43750**	42.09845	176.63655	**6059.71129**

Across all three engineering design optimisation problems, CAEPSO manages to become the best performing algorithm in four occasions, two in overall output and two in design variables. Meanwhile, DPPSO manages to become the best performing algorithm in six occasions (two in overall output and four in design variables). DACPSO accomplishes three best performing method but all in design variables.

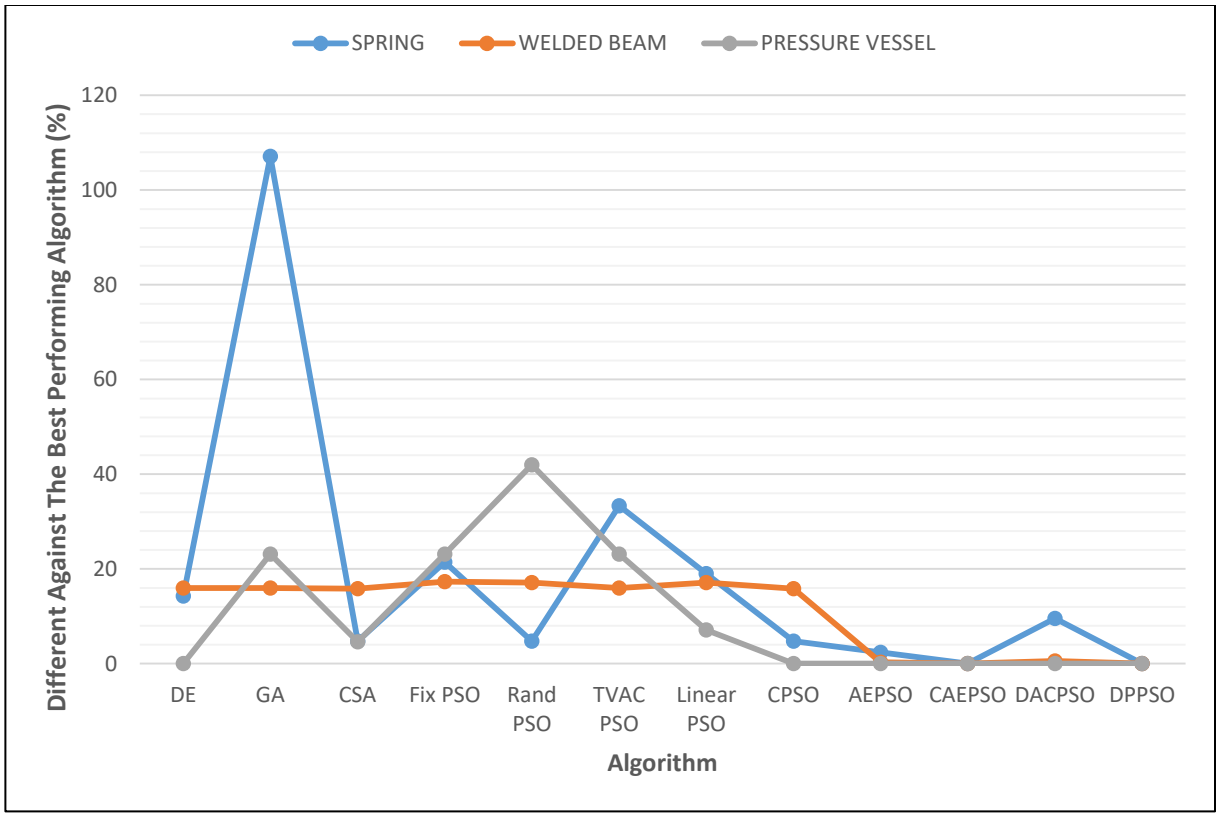


Figure 5-6. Comparison Graph of All Algorithms against the Best Performing Algorithm.

Based on observation in Figure 5-6 (comparison between algorithms against the best performing algorithm), the best method for all EDP considered is DPPSO. CAEPSO recorded almost similar results to DPPSO with the different between them is all less than one percent. AEPSO at the second place with less than two percent. However, GA recorded the worst result in spring design problem with the different against DPPSO more than hundred percent. For welded beam design optimisation problem, all algorithms are shown almost twenty percent different against DPPSO except for AEPSO, CAEPSO, and DACPSO.

5.6 Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation

The same statistical significance analysis from the previous experiment is applied to the results obtained in this experiments. The statistical tools use is N-Way ANOVA and Kruskal-Wallis with Lilliefors test is applied before to define the parametric nature of the results. The statistical significance is evaluated on benchmark functions and evolutionary approaches. For this Engineering Design Problems, all results from Table 5-5, Table 5-6, and Table 5-7 are considered for analysis of statistical differences.

Table 5-8. Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation (DAPSO).

Category	Benchmark Functions	Evolutionary Methods	Benchmark Functions & Evolutionary Methods
Fitness Value	$p = 0$	$p = 0$	$p = 0$

From the statistical analysis of the results, the outcomes show the occurrence of statistical significance between the performance achieved from three engineering design problems ($p = 0 < 0.5$), a number of evolutionary approaches ($p = 0 < 0.5$), and the interactions of those three engineering design problems against evolutionary approaches ($p = 0 < 0.05$). The result indicates the existence of significance different between tension/compression design problem against welded beam and pressure vessel design problems.

CAEPSO and DPPSO show clear statistical significance against GA, Fix PSO, Rand PSO, and TVAC PSO but a lack of significance difference against DE, CSA, Linear PSO, CPSO, AEPSO, and DACPSO. DACPSO indicates the statistical significance against Fix PSO and Rand PSO as illustrated in Figure 5-7.

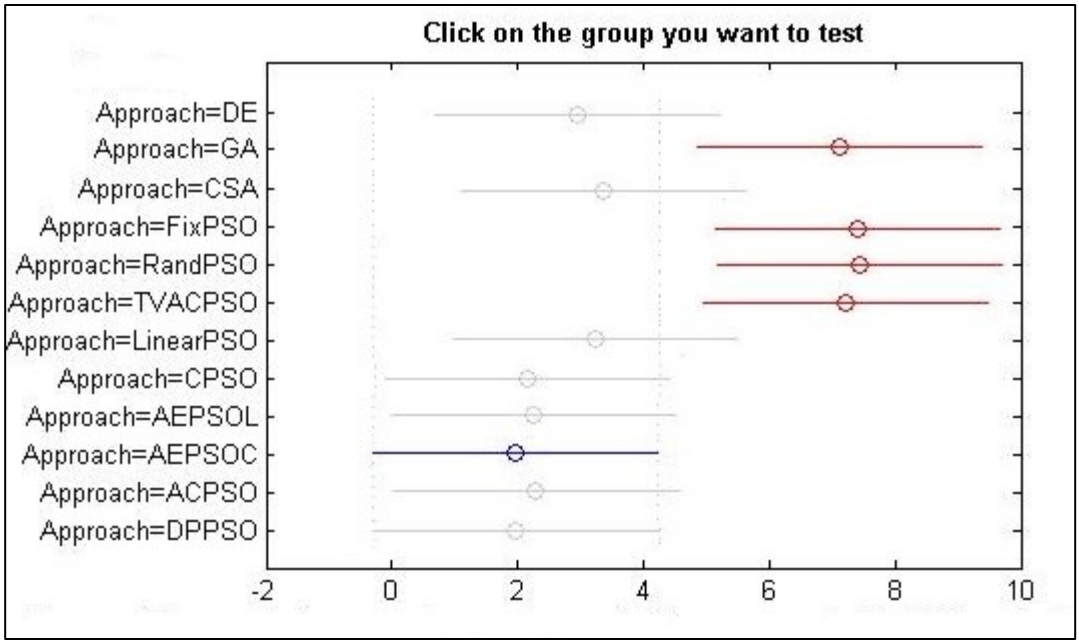


Figure 5-7. Box Plot of Significant Difference for Engineering Design Optimisation Problems (DAPSO).

5.7 Summary

This chapter discusses the results obtained for each considered evolutionary algorithms in three different types of engineering design optimisation problems. From the observation, discussion, and analysis, all proposed PSOs are considered as promising especially for MPSO, CAEPSO, and DPPSO where they managed to outperform other evolutionary algorithms quite comfortably. Dynamic approaches PSO are outstanding in welded beam design problem where their final outcomes are obviously different from the other algorithms except AEPSO. The next chapter will discuss the results of online application which is the main experiment of this research. The details of the results, discussion, and analysis for mobile robot navigation can be found in the following chapter.

CHAPTER 6

MOBILE ROBOT NAVIGATION PROBLEM (MAZE LAYOUT)

6.1 Introduction

The algorithms considered for this experiment are Fix PSO (Fix PSO), Linear Decreasing Inertia Weight PSO (Linear PSO), Constricted PSO (CPSO), Area Extended PSO (AEPSO), Constricted Area Extended PSO (CAEPSO), Dynamic Acceleration Coefficients PSO (DACPSO), and Dynamic Parameterisation PSO (DPPSO). The traditional methods for path planning are also included in this experiment which is Dijkstra's Algorithm (DA), Potential Field (PF), Rapidly-Exploring Random Tree (RRT) and Probabilistic Road Map (PRM). The Robot Operating System (ROS) is used as the platform to program the source code for all methods and embedded into a mobile robot which is discussed in chapter 2.

Seven factors are considered to assess the performance of the selected algorithms in these experiments. The first factor considered is *Time* which the total time is taken for the robot to travel from the starting location to the goal location. The next factor considered is the *Number of Collisions* where the number of collisions occurred during the runs counted. *Arrived at Destination* where the ability of the algorithm to successfully drive the robot to the goal location depending on odometry are tested. *Travelled Distance* is one of the crucial factor considered where the total distance that the robot travelled from the starting location to the goal location is evaluated. Another important factor is *Battery Consumption*, where the total of the percentage

of battery consumed on average by the robot to complete the task assigned. The data of battery consumption is solely measured from the mobile robot's battery and nothing to do with the battery consumed on computational while executing the algorithms. *Displacement Problem* is also measured as one of the factors where if the robot's final position is out of the acceptable range of the target points, then it is considered as a displacement problem. The tolerance range is one and a half size of the mobile robot used which is 90cm radius from the centre of exact coordinate location.

6.2 The Maze Layout

For this mobile robot indoor path planning experiment, a single robot (Turtlebot) is tasked to navigate through the obstacles from a fixed starting point towards a fixed destination location through a maze layout. From the various classical motion planning approaches discussed, Potential Field (PF), Dijkstra's Algorithm (DA), Rapidly-exploring Random Tree (RRT) and Probabilistic Road Map (PRM) are considered, with the understanding that detailed information of the environment is provided only for DA while all other methods rely solely on their online sensory perception. Each node in the DA represents a centroid location between two nearby obstacles and the distance between these nodes are considered as weights. Amongst the evolutionary-based motion planning approaches, GA, DE, CSA and variation of PSO are utilised. The first one is a maze layout and the second one is a symmetric layout. Both layouts are complex with obstacles rich and hard to manoeuvre around. Both layouts were run and repeated ten times for each algorithm.

The layout is appropriately named as a maze layout because of the placement of the obstacles that resulted in the robot being constantly surrounded by them whilst it is moving

towards the goal position. Figure 6-1 illustrates the layout of the testing environment in this experiment. The starting location (marked as S), target location (marked as D) and obstacles are represented in blue, red and black colours respectively. The obstacles are rectangular in shape and they all have the same dimensions (except for two obstacles that are marked as B).

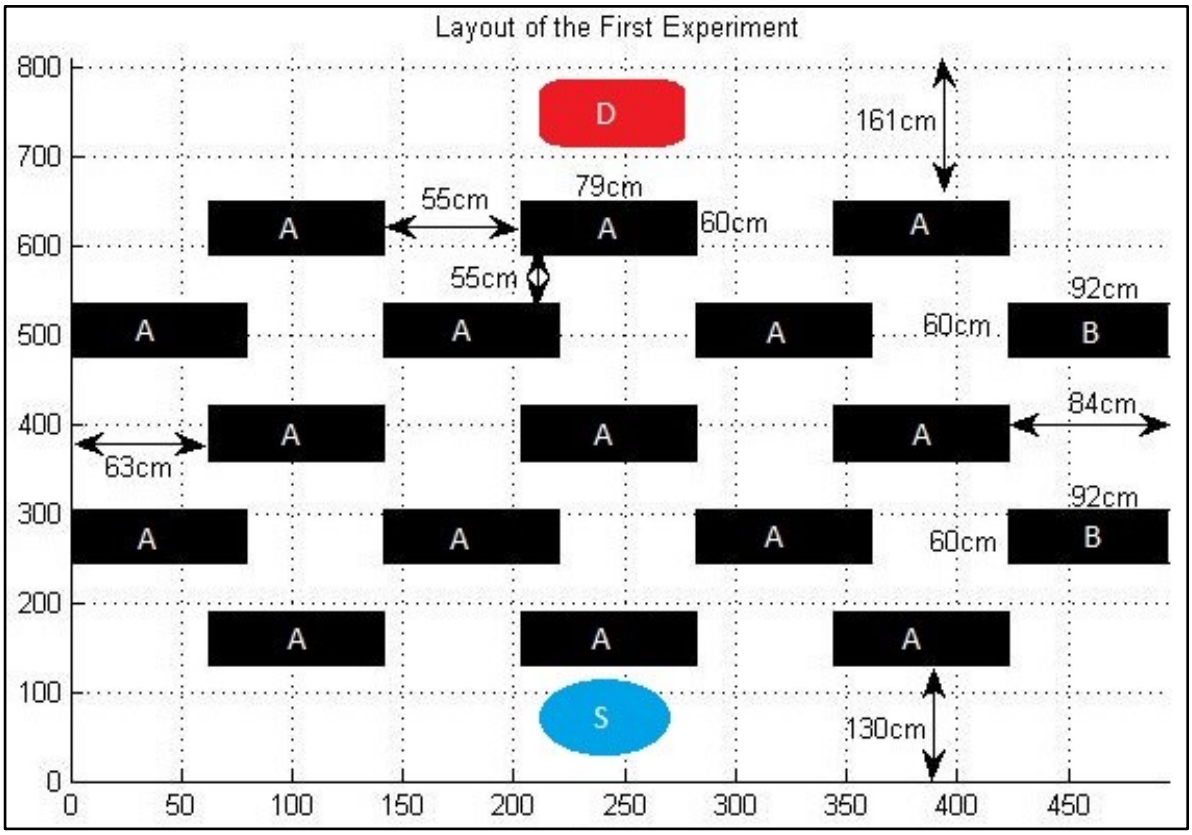


Figure 6-1. The Maze Layout.

The dimensions of the obstacles marked as A in the above figure are 79cm × 60cm and the other two obstacles, marked as B, are 92cm × 60cm. The gaps between the obstacles are consistent throughout the layout and are set to be less than twice of the size of the robot (55cm). The gaps between the obstacles and the walls are set to 63cm on left and 84cm on the right. The

dimensions of the maze layout are set to 8m × 5m. Figure 6-2 offers four snapshots from different angles of the maze layout from which the density of the obstacles is evidenced.

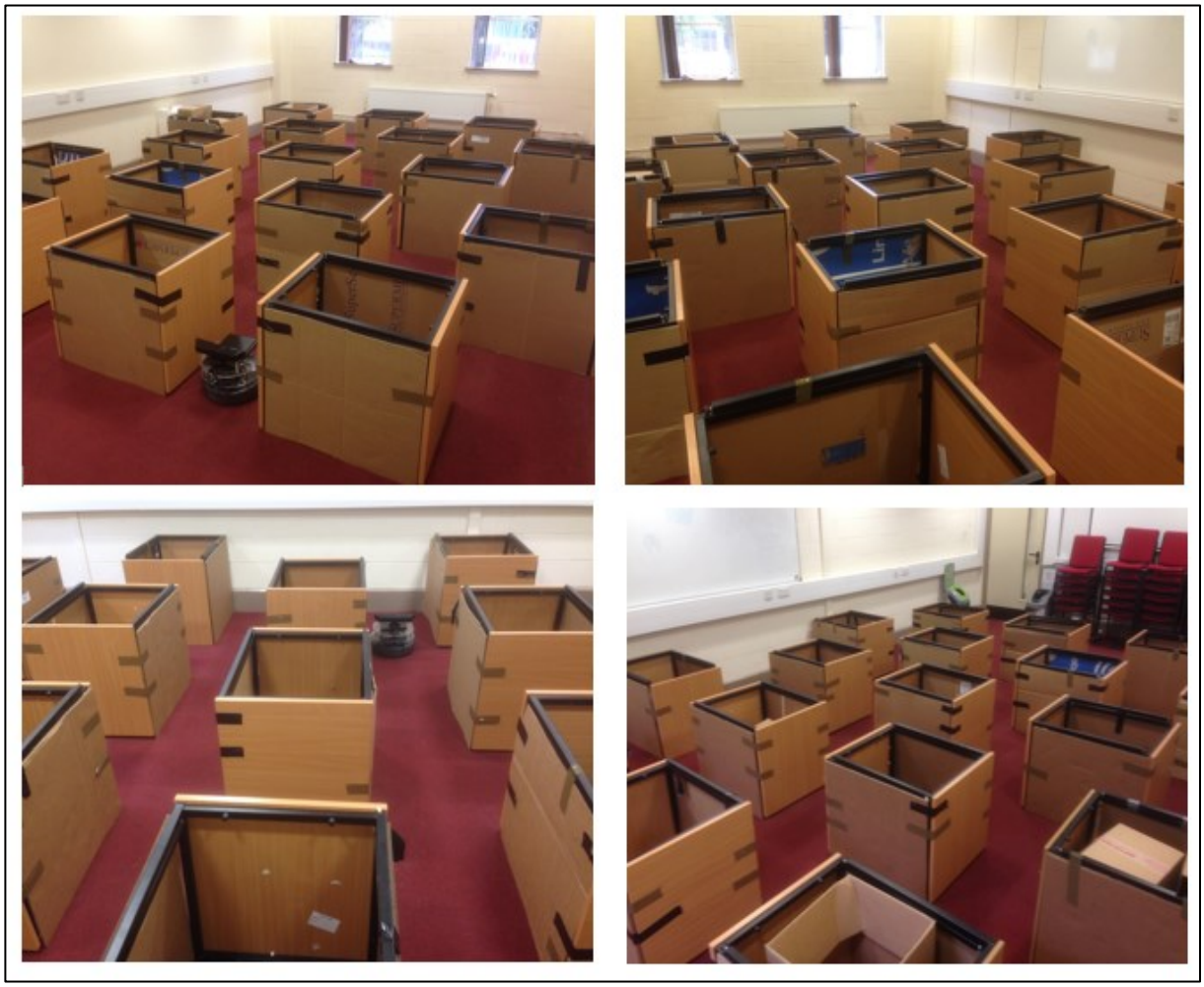


Figure 6-2. The view of Path Planning Experiment (Maze Layout) from Four Difference Angles.

The Turtlebot robot platform (as shown in Figure 6-3) with a width of 30 cm is utilised in combination with the Robotic Operating System (ROS) to carry out all of the path planning experiments involved in this research. Two touch sensors and six infrared sensors are utilised as the main sensing equipment. In order to protect the robot from having a high-speed collision that can cause a significant amount of error in the odometry sensor data, the robot is instructed

to immediately reduce its velocity once the infrareds detect any objects within its range. Hence, any collision that occurs is considered as a controlled collision and the robot is programmed to execute an immediate stop and reverse mode to prevent a distortion of the odometry sensor data. The procedure to address the required recovery steps from the obstacle collision includes i) choosing a temporary destination depending on obstacle position, ii) turning towards a temporary destination, iii) moving towards the temporary target location for 2s, iv) returning the control to the motion planner algorithm.

Figure 6-3. The Turtlebot.

6.3 Result for Morphology Particle Swarm Optimisation

This subsection discussed the results, performance and significance analysis for Morphology PSO in both two layouts involved in path planning for mobile robot experiment. The first results are based on Maze Layout with only one starting point and one target point. Meanwhile, the second results from Symmetric Layout consist of three sub-experiments with a combination of two starting points and two target points.

The average results of MPSO and CMPSO against other evolutionary algorithms and motion planning algorithms achieved from ten executions in mobile robot navigation experiment for maze layout are reported Table 6-1. The results are discussed based on the factors considered above. From the observations, all algorithms managed to find a route between the starting and the destination points. All these algorithms also managed to avoid any collision with the obstacles within the maze environment. Within the ten runs executed, all algorithms successfully arrived at the destination without any displacement problem. These due to the same mechanism of obstacle avoidance and odometry calculation used for all algorithms.

Table 6-1. MPSO Result for Path Planning in Maze Layout

Algorithm	Arrived at Destination (Times)	Number of Collisions (Times)	Displacement Problem (Times)	Execution Time (Seconds)	Battery Consumption (%)	Travelled Distance (Meter)	Convergence Iterations	Best Performing Algorithm (Out of 6/7)
PF	**10**	**0**	**0**	287.5688	2.7671	12.8936	-	3
DA	**10**	**0**	**0**	**255.6497**	**1.5505**	**11.8966**	-	6
RRT	**10**	**0**	**0**	466.7081	2.8598	13.7719	-	3
PRM	**10**	**0**	**0**	574.9445	4.6810	14.4165	-	3
DE	**10**	**0**	**0**	257.4400	1.5874	12.1954	542.0	3
GA	**10**	**0**	**0**	323.4239	1.9770	12.7922	565.3	3
CSA	**10**	**0**	**0**	306.6856	1.8880	12.1075	550.7	3
Fix PSO	**10**	**0**	**0**	273.2990	4.5097	12.2378	551.0	3
Rand PSO	**10**	**0**	**0**	296.9488	1.7990	12.2994	548.1	3
TVAC PSO	**10**	**0**	**0**	299.8347	1.8064	12.1073	531.9	3
Linear PSO	**10**	**0**	**0**	307.3298	1.8398	12.3441	544.1	3
CPSO	**10**	**0**	**0**	262.6602	1.7313	12.1088	544.5	3
MPSO	**10**	**0**	**0**	261.5600	1.5838	12.0148	**508.9**	4
CMPSO	**10**	**0**	**0**	263.2141	1.5683	12.1733	517.6	3

Concerning execution time factor, DA managed to become the best performing algorithm with 255.6497 seconds on average to complete the navigation task. This performance is closely followed by MPSO and CPSO with 261.5600 seconds and 262.6602 seconds as second and third best performing algorithm respectively. In the battery consumption category, DA managed to use only 1.5505% power consumption on average and become the best performing algorithm. However, CMPSO is slightly short from being the best performing algorithm with an average of 1.5683% battery consumption.

MPSO and DE are the third and fourth best performing algorithm with an average of 1.5838% and 1.5874% battery consumption. DA recorded an average of 11.8966 meters in total travelled distance category to outperform others. MPSO come as second best performing algorithm with an average travelled distance of 12.0148 meters. This result is closely followed by CPSO and CMPSO with an average travelled distance of 12.0148 and 12.1733 meter to become the third and fourth best performing algorithm. Convergence Iteration category is only applied to evolutionary based algorithms.

In this category, MPSO managed to outperform other EA-based methods with an average of 508.9 iterations in ten runs. CMPSO and TVAC PSO are the second and third performing methods with 517.6 and 531.9 average iterations per execution respectively. Regarding being the best performing algorithm, DA is the most outstanding algorithm with being the best performing algorithm in each category considered except convergence iterations category which dominated by MPSO.

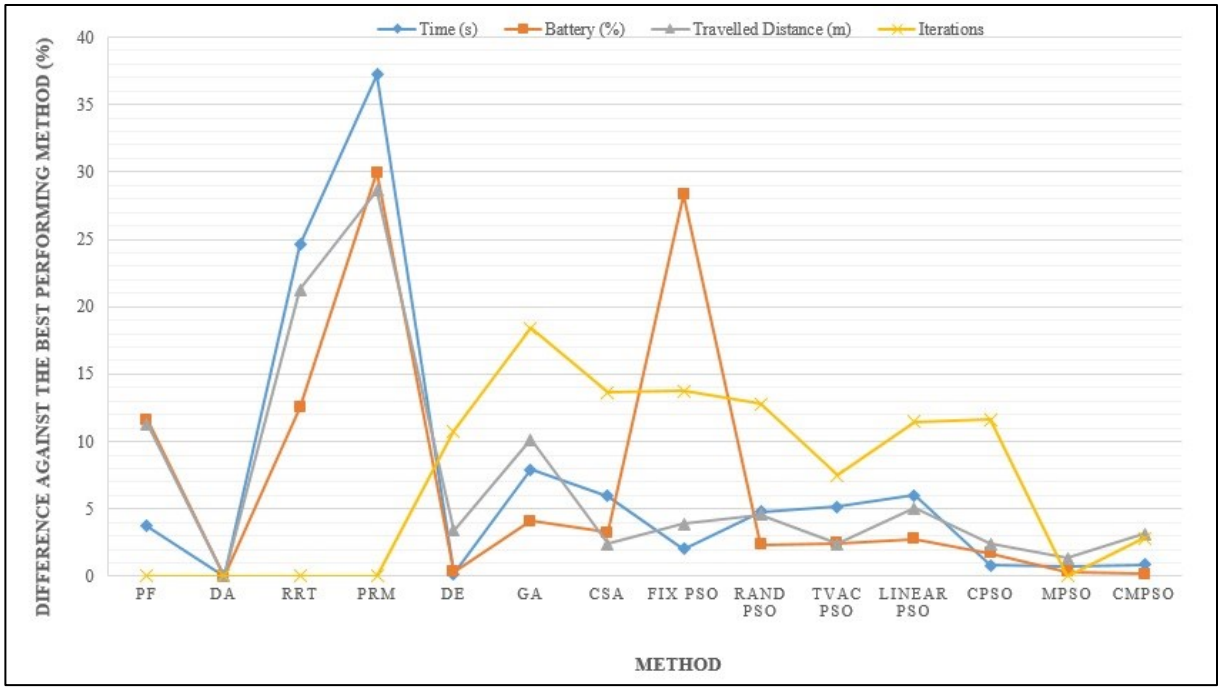


Figure 6-4. The graph of the best performing algorithm against other algorithms (For All Factors).

Figure 6-4 shown the performance of all algorithms against the best performing algorithm in four factors considered (Execution Time, Battery Consumption, Travelled Distance and Numbers of Iterations). DA is the best performing algorithm for all factors (with omission for iteration factor) but considering its superiority in term of map information, it is safe to state that MPSO is the closest local path planning performing as decent as DA for all factors. DE is also showing quite close performance in 2 out of 3 factors considered against DA with CPSO is more consistent throughout all three factors.

Figure 6-5 illustrated the trajectory traces for all utilised algorithms in mobile robot navigation problems for maze layout. Each sub-figure in Figure 6-5 represents ten paths travelled by the Turtlebot robot using a specific motion planning and navigation technique. These ten routes in each sub-figure are illustrated using different colour codes. From the observation on the figure, DA stands out as the most consistent algorithm. MPSO, CMPSO,

DE, GA, CSA, and all variants of PSO trajectory traces are almost similar between one another where they have shown high precisions with only one or two executions not following the same trajectory (only towards the end of the journeys). The trajectory traces of PF, RRT, and PRM indicate a lack of consistency and low precision since almost none of the travelled paths are similar. It is due to the EA-based algorithm using almost the same mechanism which is dealing with obstacles and feasible paths or routes depending on their current position.

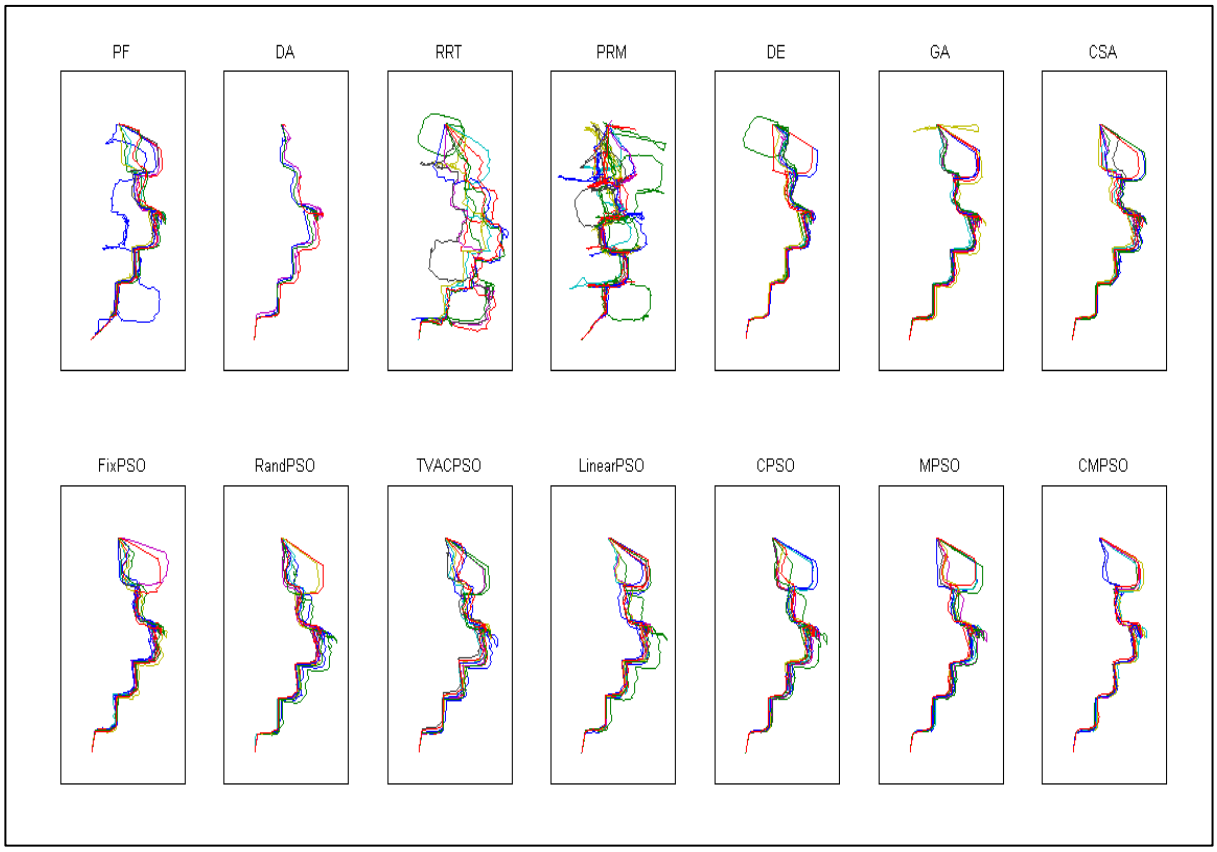


Figure 6-5. Trajectory traces of employed approaches (including Morphology based PSO) in Path Planning Experiment for Maze Layout. Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.

In order to better understand the reasoning behind the performance differences between various approaches utilised in the research, a unique representation of the robot's motion is used.

In this representation, the experiment layout is divided into nine equal-sized rectangles; each called a region.

The average number of times (across ten executions) that the robot is located in each region is recorded. The results are presented in the form of pie charts in Figure 6-6. In this experiment, regions 2 and 9 contains starting and end points respectively. It is noteworthy that DA, as the best performing approach, only considered regions 2, 3, 6 and 9 in its trajectories with regions 6 and 3 having the highest and the least contributions in robot's chosen trajectory respectively. Given that regions 4, 5, and 6 are the regions containing three parallel rows of obstacles, these areas are likely to be the most difficult to manoeuver. This issue is reflected in the pie charts illustrated in Figure 6-6 in which the highest percentage is dedicated to region 6. This is with exception of RRT algorithm in which region 3 received the highest percentage of occupation. MPSO and CMPSO behaviour are almost similar to DA in term of the region occupied with only slight different of percentage amongst four regions occupied.

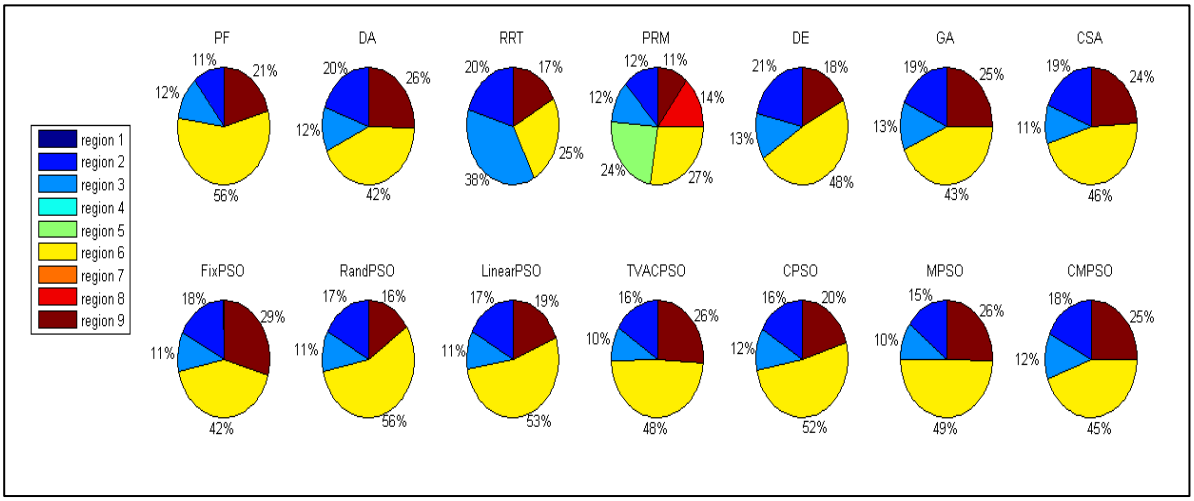


Figure 6-6. Region Occupied for Path Planning Experiment (Maze Layout).

It is also noticeable that in the case of PRM algorithm the occupation percentage observed in region 6 of other methods is shared between regions 5 and 6. Unlike other approaches, PRM also included region 8 in its trajectories. The inclusion of these additional regions (5 and 8) are likely to be the reason behind the poor performance achieved by PRM algorithm in this experiment. RRT, unlike PRM, did not include any additional regions in its trajectory towards the destination, however, from the results in the pie chart, it is noticeable that this approach spent an unusual amount of time for manoeuvring within region 3.

It is obvious by allocating 39% to region 3 by RRT in comparison with DE and GA in which the robot occupied this region in only 12% of its trajectory toward the destination. The second best performing algorithm (MPSO) reported a smaller percentage (15%) for occupying region 2 compared with DA (20%). Furthermore, in comparison to DA, MPSO also reported a smaller percentage in regions 9 (destination region) while it occupied most of its time in region 6 with 49%. Similar performance is observed by PF which reported 56% average occupation of region 6 while outmanoeuvring DA and MPSO in region 2 with 11% average occupation. The differences observed between MPSO and DA in their manoeuvrability in region 6 can be due to DA's advantage in terms of having full knowledge of the environment layout.

6.4 Significance Analysis for Morphology Particle Swarm Optimisation

Table 6-2. Statistical Analysis for Morphology Particle Swarm Optimisation (MPSO).

Category	Experiments	Approaches	Experiments & Approaches
Time (s)	-	p = 6.44912e-29	-
Battery Consumption (%)	-	p = 1.70000e-02	-

Travelled Distance (m)	-	p = 4.87912e-22	-
Convergence iteration	-	p = 1.70000e-02	-

Similar to experiments in chapter 5 and 6, statistical analysis of the results is considered to further assess the findings in this motion planning experiment for maze and symmetric layouts and the results are shown in Table 6-2. The analysis is performed based on four categories of battery consumption, execution time, travelled distance and convergence iteration. Following observations are made for the path planning experiment using maze layout:

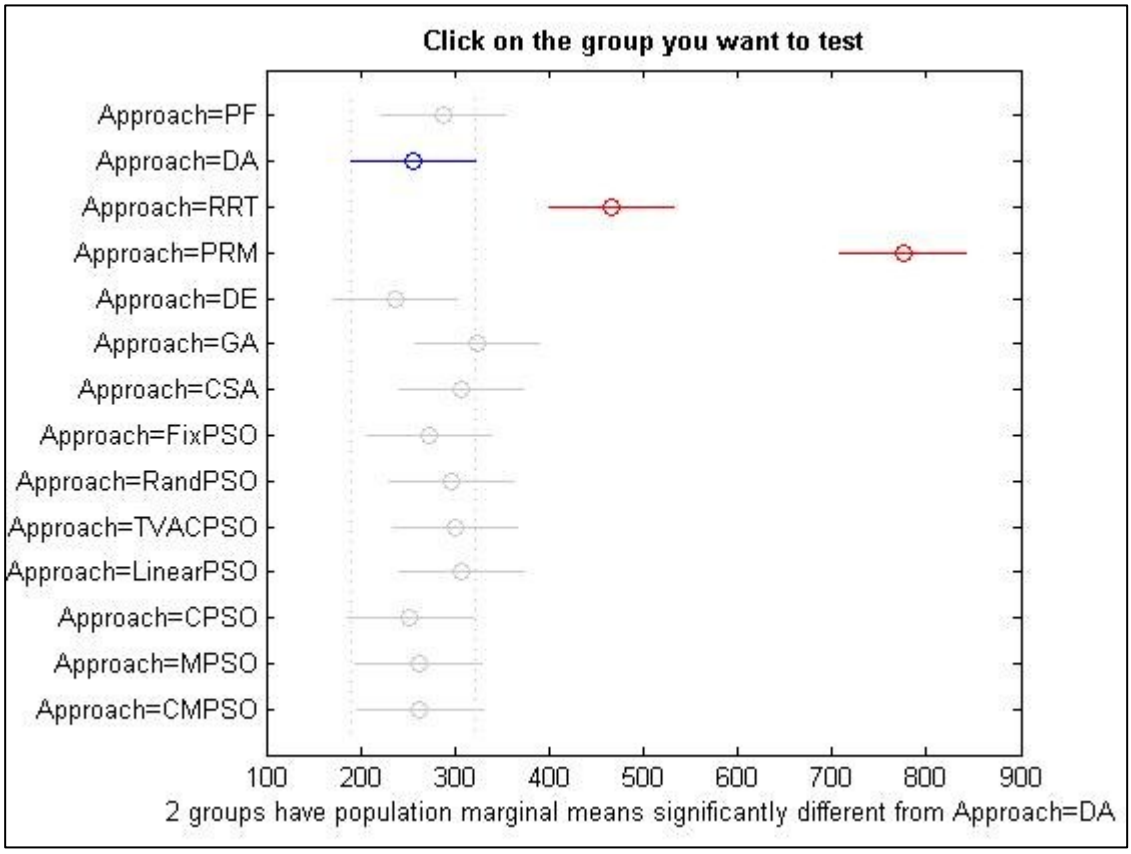


Figure 6-7. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Execution Time factor.

- Execution time (*s*): The results of statistical significant analysis indicated such significant in approaches (p = 4.87912e-22 < 0.05). Based on the observation from Figure 6-7, RRT and PRM are found to be a statistically significant difference from all other approaches and each other. However, there is no statistically significant difference observed between PF, DA, DE, GA, CSA, Fix PSO, Rand PSO, TVAC PSO, Linear PSO, CPSO, MPSO, and CMPSO.

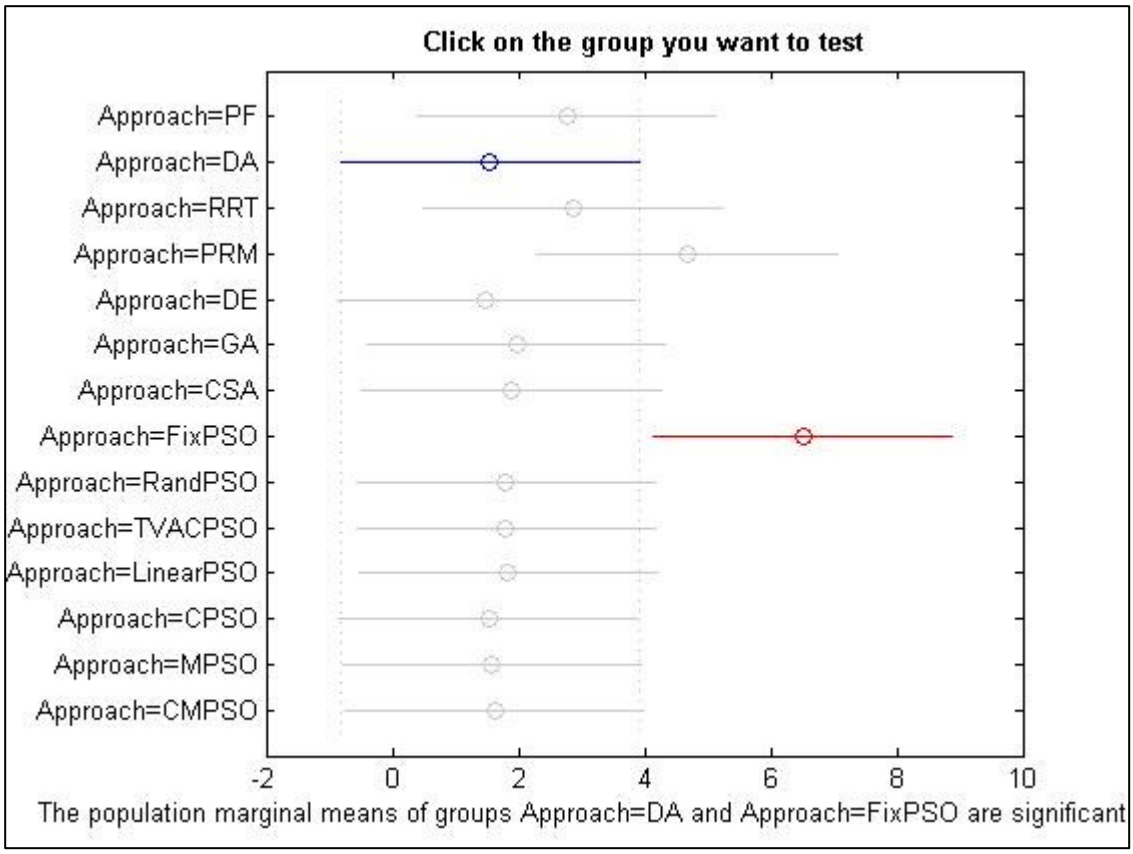


Figure 6-8. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Battery Consumption factor.

- Battery Consumption (*%*): Statistical significant analysis of the results indicated existent of such significant among the approaches utilized (p = 6.4491e-29 < 0.05). From the methods comparison, statistical significant analysis indicated a lack of

significant difference between each other except for Fix PSO which shown significant difference against DA, DE, CPSO, MPSO, and CMPSO as illustrated in Figure 6-8.

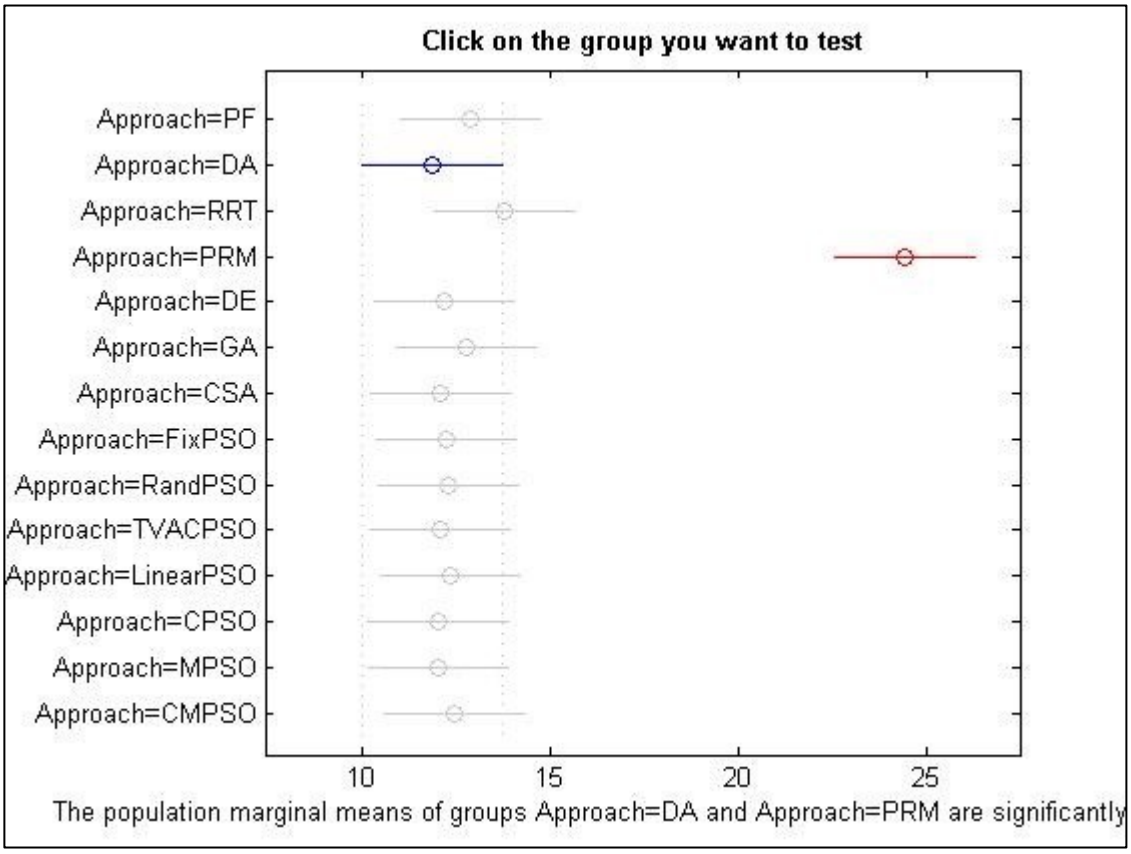


Figure 6-9. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Travelled Distance factor.

- The length of the travelled path (m): Statistical significant analysis of the results indicated existent of such significant among the approaches utilized (p = 4.87912e-22 < 0.05). Within this category, only PRM once indicated significant difference from others while the rest indicated a lack of statistical significant amongst themselves.

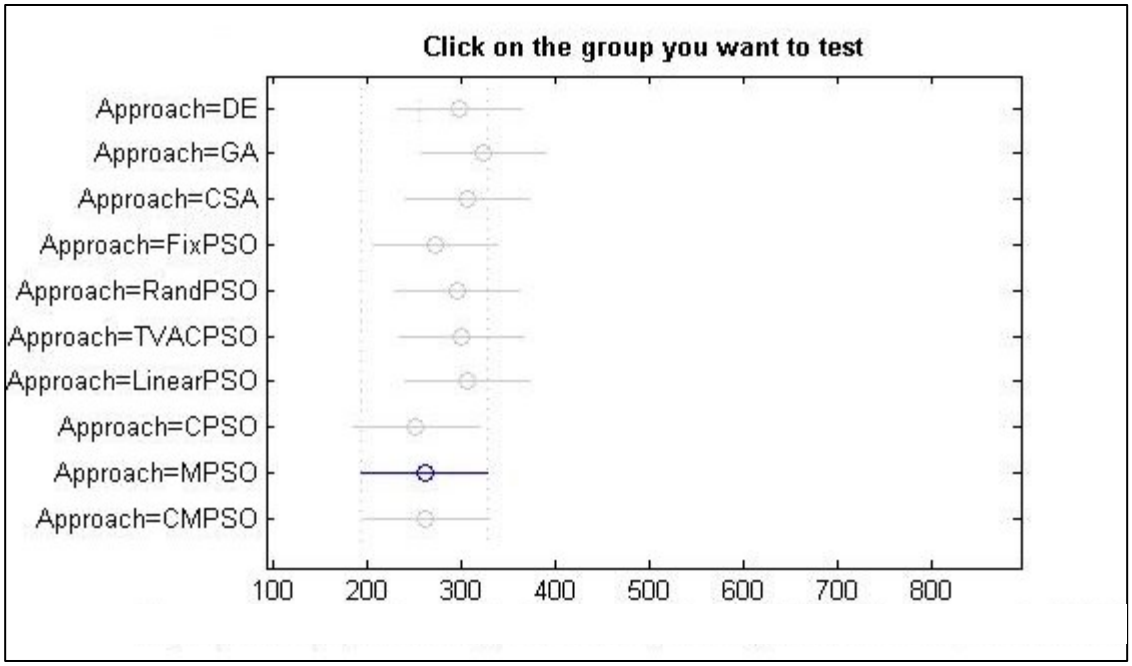


Figure 6-10. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Convergence Iteration factor.

- Convergence iteration: The statistical analysis of the results indicate a lack of significant in approaches ($p = 0.17 > 0.05$). As seen in Figure 6-10, there is no significant different between approaches.

6.5 Result for Dynamic Approaches of Particle Swarm Optimisation

This subsection discussed the results, performance and significance analysis for Constricted Area Extended PSO (CAEPSO), Dynamic Acceleration Coefficients PSO (DACPSO) and Dynamic Parameterising PSO (DPPSO) in the maze and symmetric layouts for mobile robot navigation experiments. The results are divided into another two sub-section based on their layout.

The same method is used where average results achieved from ten executions of motion planning algorithms are reported in Table 6-3. The results are discussed based on the same factors discussed in previous sub-section which were number of obstacle collisions, number of runs with successful arrival to the destination point, number of runs with displacement problem, average execution time, average battery consumption, average travel distance, and average convergence iteration (excluding classical path planning methods). From the observations in Table 6-3, all algorithms managed to find a route between the starting and the destination points without any displacement problem. These are due to the same platform and obstacle behaviours applied to all algorithms involved. The different however can be observed within four remaining considered factors (execution time, battery consumption, total travelled distance, and iteration numbers.

For execution time category, DA managed to outperform other algorithms with an average time taken to complete the motion planning of 87.5493 seconds. DPPSO and CAEPSO become second and third best performing algorithm with average execution time of 120.3214 seconds and 123.4300 seconds respectively. CPSO closely follows it with 126.6746 seconds and PF with 126.7588 seconds. Next considering factor is battery consumption, where DA once again outperformed others with an average energy consumption of only 0.3524%. The second best performing method is DPPSO with an average of 0.4432% and narrowly followed by CPSO with an average of 0.4637%. CAEPSO and DACPSO reported an average battery consumption of 0.6385% and 0.5719% respectively.

Table 6-3. DAPSO Results for Path Planning in Maze Layout

Factor	Arrived at Destination	Number of Collisions	Displacement Problem	Execution Time (sec)	Battery Consumption (%)	Travelled Distance (m)	Convergence Iterations	Best Performing Algorithm
PF	**10**	**0**	**0**	287.5688	2.7671	12.8936	-	3
DA	**10**	**0**	**0**	255.6497	**1.5505**	**11.8966**	-	5
RRT	**10**	**0**	**0**	466.7081	2.8598	13.7719	-	3
PRM	**10**	**0**	**0**	574.9445	4.6810	14.4165	-	3
DE	**10**	**0**	**0**	257.4400	1.5874	12.1954	542.0	3
GA	**10**	**0**	**0**	323.4239	1.9770	12.7922	565.3	3
CSA	**10**	**0**	**0**	306.6856	1.8880	12.1075	550.7	3
Fix PSO	**10**	**0**	**0**	273.2990	4.5097	12.2378	551.0	3
Rand PSO	**10**	**0**	**0**	296.9488	1.7990	12.2994	548.1	3
TVAC PSO	**10**	**0**	**0**	299.8347	1.8064	12.1073	**531.9**	3
Linear PSO	**10**	**0**	**0**	307.3298	1.8398	12.3441	544.1	3
CPSO	**10**	**0**	**0**	262.6602	1.7313	12.1088	544.5	3
AEPSO	**10**	**0**	**0**	258.1006	1.8813	12.1370	554.1	3
CAEPSO	**10**	**0**	**0**	256.6345	1.7046	12.1051	540.0	4
DACPSO	**10**	**0**	**0**	270.4412	1.8294	12.2550	545.1	3
DPPSO	**10**	**0**	**0**	**255.1144**	1.6978	12.0125	541.2	4

In the travelled distance category, DA was outperformed by DPPSO and CAEPSO with a mean results of 7.3982 meters and 7.4333 meters respectively while DA only managed to record 7.4706 meters on average. TVAC PSO surprisingly outdone other algorithms for iteration numbers category with an average of 422.7 iterations. Rand PSO just slightly short from TVAC PSO with an average result of 422.9 iterations, followed by Fix PSO with 432.8 iterations.

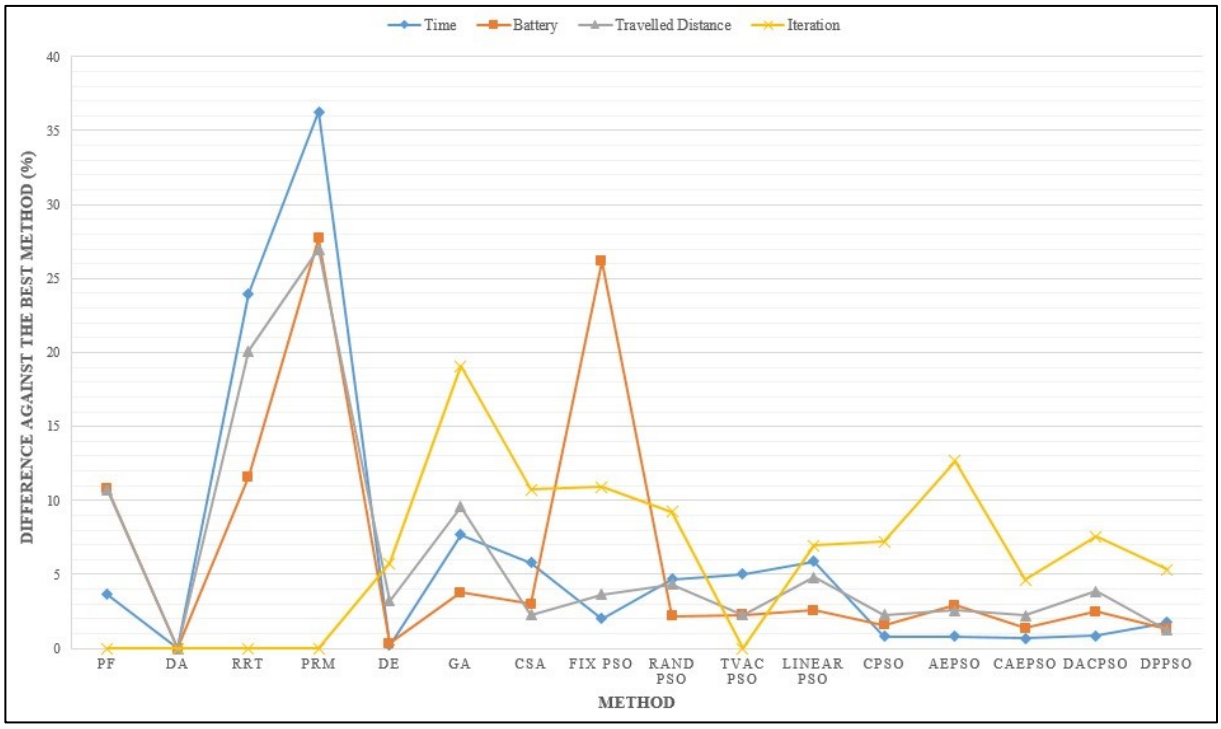


Figure 6-11. The graph of the best performing algorithm against other algorithms (For All Factors).

Figure 6-11 shown the performance of each algorithm chosen against the best performing algorithm (DA) in 4 factors (execution time, battery consumption, travelled distance and numbers of iterations). As DA has full information about the map, it is considered as the best results can be obtained by the other algorithms. From the observation on the graph, DPPSO

is the most consistent algorithm throughout all 3 factors considered against the DA. CAEPSO and CPSO shown a quite similar result but CPSO seems more consistent across three categories measured.

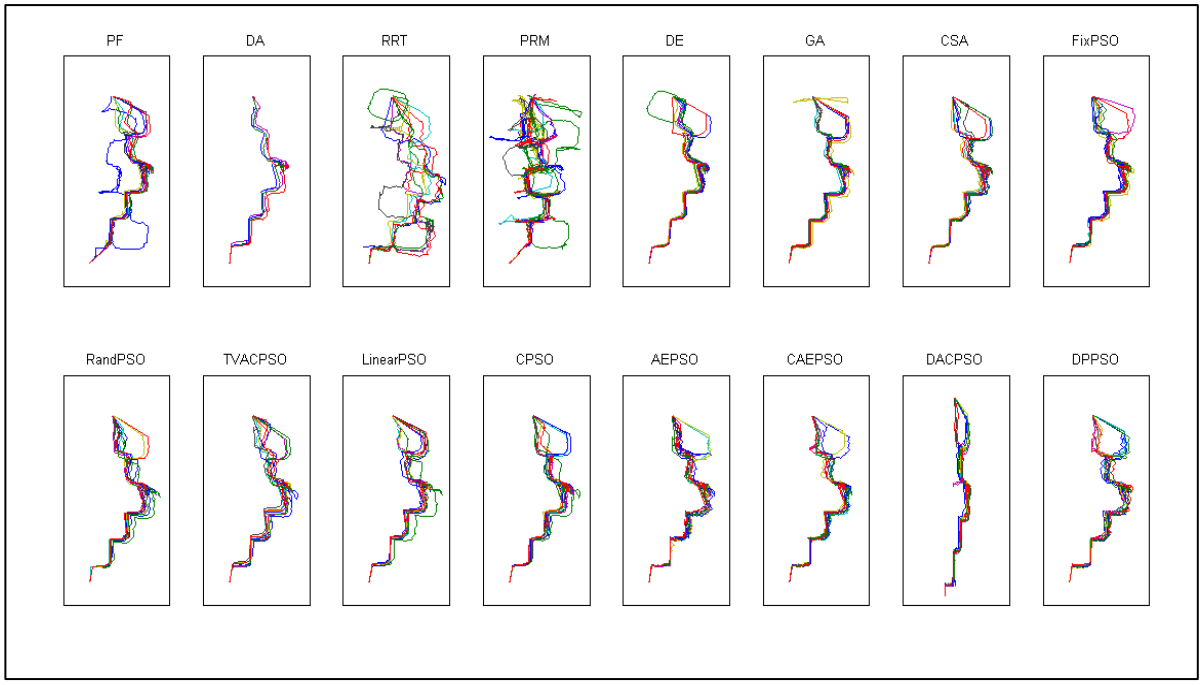


Figure 6-12. Trajectory Traces for DAPSO in Maze Layout Path Planning.

Figure 6-2 illustrated the trajectory traces for all utilised algorithms including Dynamic Approaches PSO in maze layout. Each sub-figure in Figure 6-2 represent ten paths travelled by the turtlebot robot using a specific motion planning and navigation technique. These ten routes in each sub-figure are illustrated using different colour coding. From the observation on the figure, DA, CPSO, AEPSO, CAEPSO, and DPPSO stands out as the most consistent algorithm. The trajectory traces of PF, RRT, PRM, and TVAC PSO indicate the least consistency compared to the other methods. AEPSO, CAEPSO, and DPPSO shown almost a similar trajectory traces which mean they used almost the same paths. Other algorithms that have

similar paths as AEPSO, CAEPSO, and DPPSO are Fix PSO, Rand PSO, and CPSO. DE and GA also have similar path with one to two paths shown their wandering behaviour.

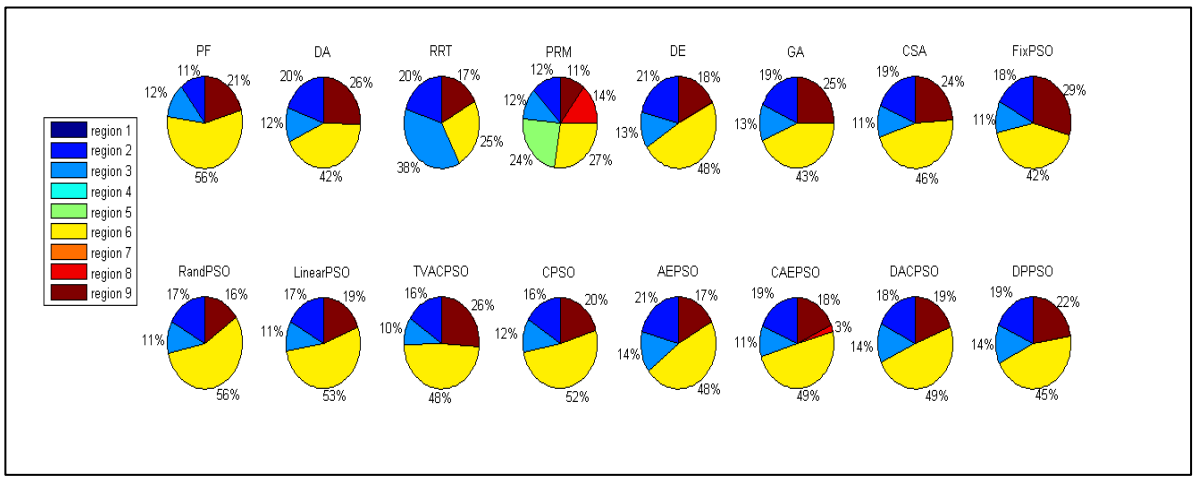


Figure 6-13. Pie Chart of Region Occupied for Dynamic Approaches PSO against other algorithms in Maze Layout.

In order to have better understanding about the performance differences among various approaches utilized in the research, a unique representation of the robot's motion is presented. In this presentation, the experiment layout is divided into nine equal-sized rectangles, each rectangle known as a region and the average number of times (across ten executions) that the robot is positioned in each region is noted. The results are illustrated in the form of pie charts as seen Figure 6-13. It should be noted that given that there are fundamental differences between the performances of some of the approaches considered in the research and in order to be able to compare the findings presented in pie charts form, the results are rescaled to the range of 1 to 100. For this layout, the starting point is within region 2 and the destination point is within region 9.

Considering regions 4, 5, and 6 are the regions consisting three parallel rows of obstacles, these areas are likely to be the hardest to maneuverer. This issue reflected itself in pie charts illustrated in Figure 6-13 in where all methods consist the highest percentage in region 6 (with the exception of RRT). The best performing approach, DA, only used regions 2, 3, 6, and 9 in its trajectories with regions 6 and 3 having the highest and the least contributions in robot's chosen trajectories respectively. All algorithms is considering the same regions as DA except for PRM where it shown a sign of struggled to minimise the number of regions occupied. This could be a reason for PRM to not perform well although eventually PRM did manage to find the destination point. CAEPSO did consider a small fraction of region 8 (3% occupation) in its trajectories due to destination point location in region 9 but quite close to region 8. CAEPSO, DACPSO, and DPPSO have recorded close percentage occupation to DA. For region 1, CAEPSO and DPPSO used 19% of their time in region 1 while DACPSO occupied region 1 with 1% shorter time compared to those two. For region 3, DACPSO and DPPSO shared the same occupation percentage (14%), and CAEPSO was occupying the region for 11%. CAEPSO and DACPSO spent the same amount of time navigating region 6 with 49% while DPPSO used 5% lesser time in region 6.

6.6 Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation

Table 6-4. Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation (DAPSO)

Category	Experiments	Approaches	Experiments & Approaches
Time (s)	-	p = 5.13884e-34	-
Battery Consumption (%)	-	p = 1.23000e-02	-

Travelled Distance (m)	-	p = 6.27720e-24	-
Convergence iteration	-	p = 7.23100e-01	-

Similar to statistical analysis done for Morphology PSO for mobile robot navigation experiments in previous sub-section, the analysis is performed based on four categories of battery consumption, execution time, travelled distance, and convergence iteration (Table 6-4). The following results are seen:

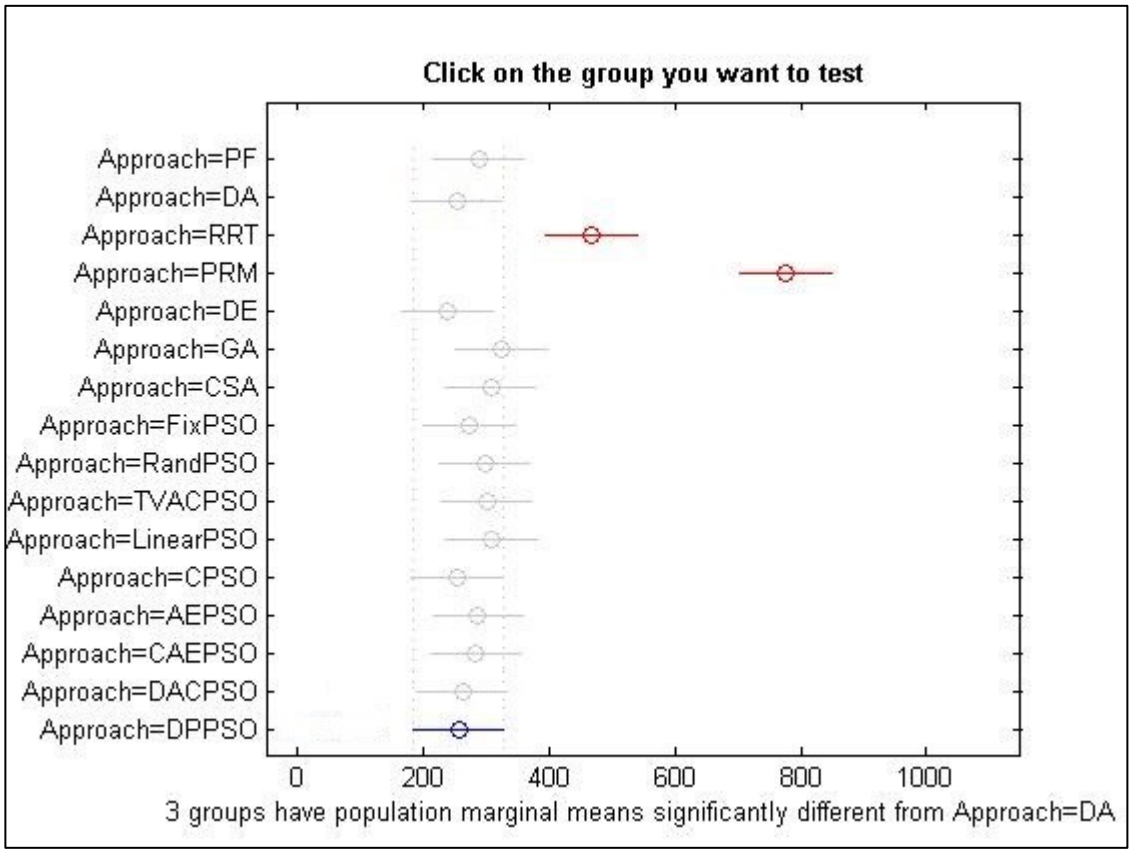


Figure 6-14. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Execution Time factor.

- Execution time (s): The statistical analysis shown significant difference between approaches applied (p = 5.13884e-34 < 0.05). The significant difference is observed

(based on Figure 6-14) between all algorithms against RRT and PRM with the exception of GA. GA is only shown significant difference against PRM.

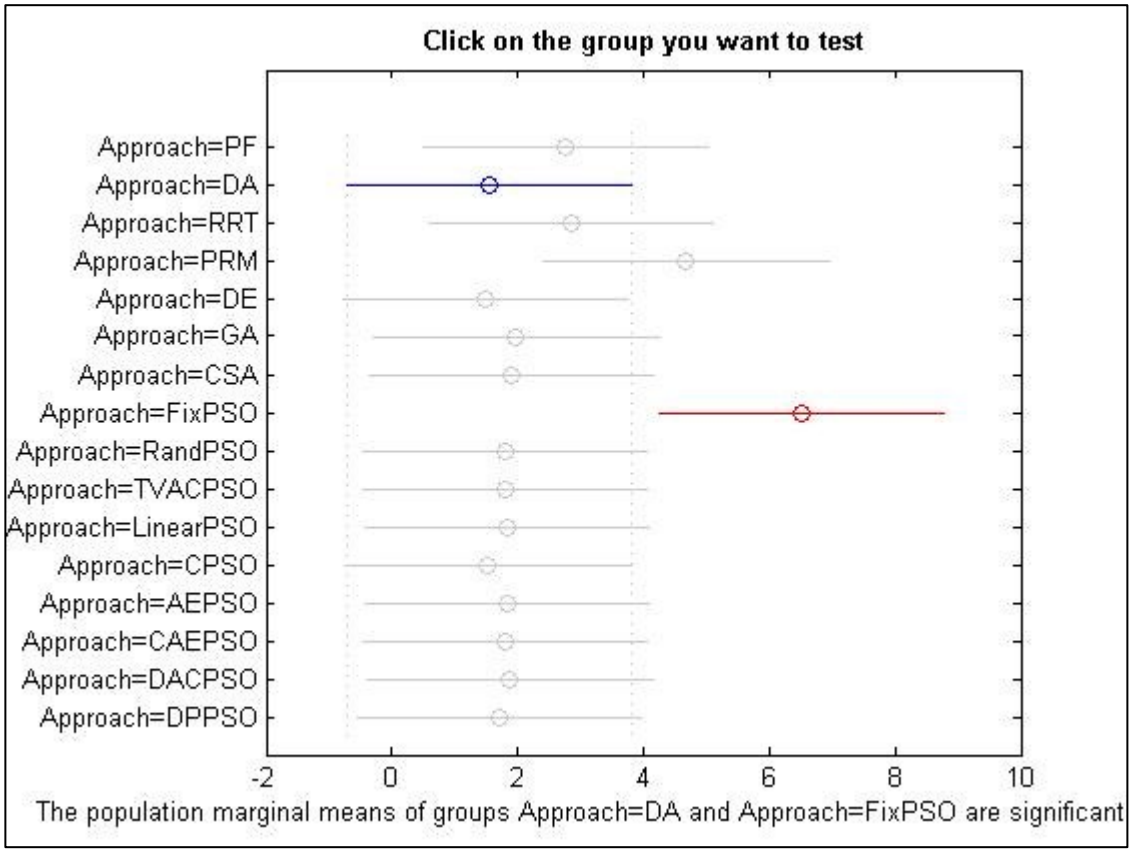


Figure 6-15. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Battery Consumption factor.

- Battery Consumption (*%*): The outcome of statistical analysis recorded significant difference between approaches (p = 1.23000e-02 < 0.05). Only PF, RRT, and PRM showed lack significant difference against the worst performing algorithm for this category, Fix PSO. The rest of algorithms shown significant different against Fix PSO and not between them.

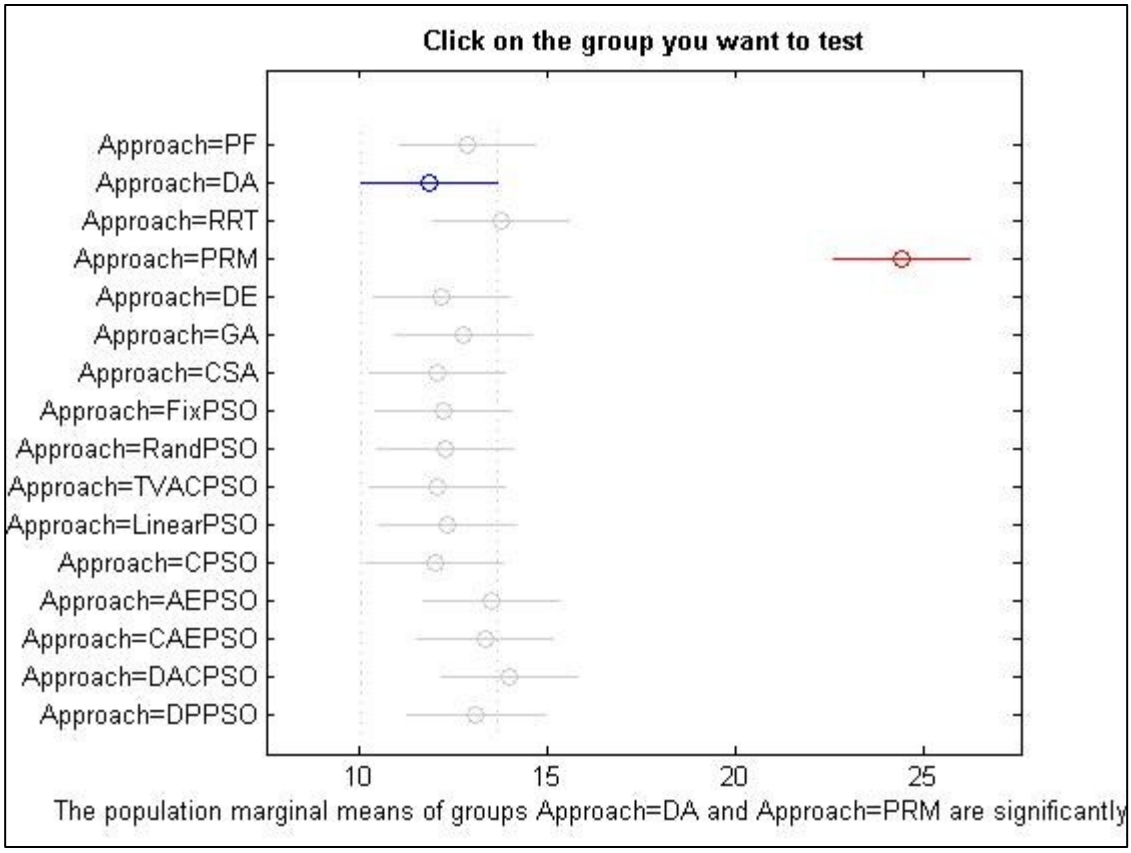


Figure 6-16. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Travelled Distance factor.

- Total travelled distance (*m*): Although, the results of the statistical significant analysis indicated such significant amongst methods (p = 6.2772e-24 < 0.05) but only PRM showed significant difference between other methods. Meanwhile, other methods indicated a lack of significant difference amongst themselves.

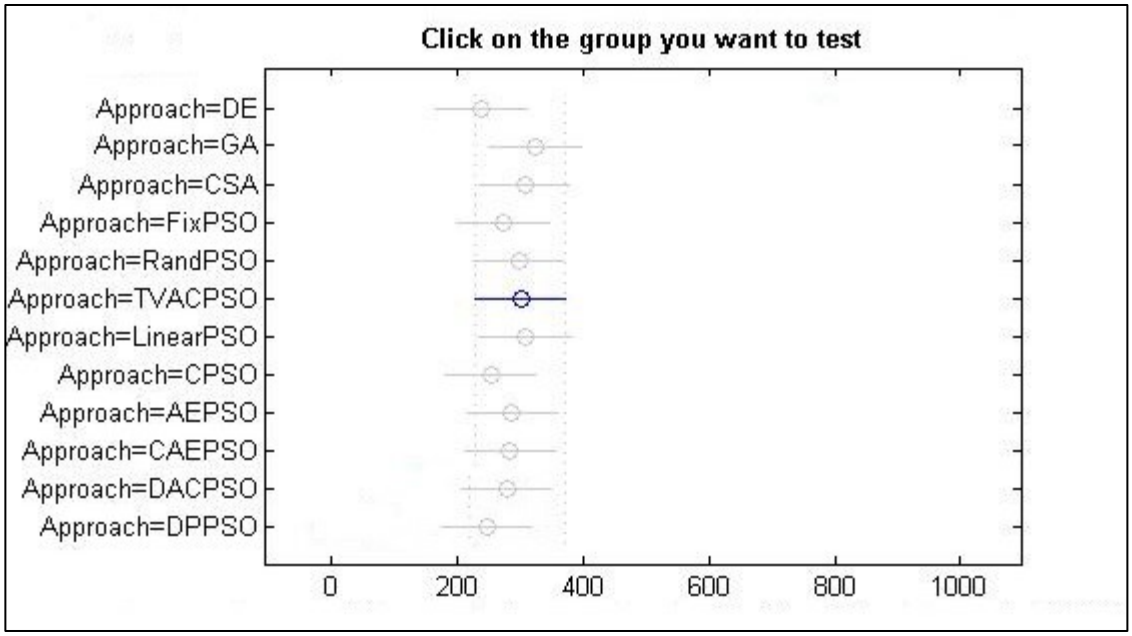


Figure 6-17. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Maze Layout) for Iterations factor.

- Convergence iteration: The results of statistical analysis shown a lack of significant difference between algorithms ($p = 0.7231 > 0.05$). None of the algorithms shown any significant difference between themselves.

6.7 Summary

This chapter has discussed the results achieved from two set of layouts for mobile robot navigation experiment. Two types of proposed PSO (Morphology based PSO and Dynamic Behaviours based PSO) were tested against twelve algorithms and thirteen algorithms respectively (including four classical path planning methods). From the observation, discussion, and analysis, all newly introduced PSOs are considered as encouraging especially for MPSO, CAEPSO, and DPPSO where they managed to outperform other evolutionary algorithms (with the exception of DA, due to its advantages) almost easily. These three algorithms are

outstanding especially sub-experiment 1 for symmetric layout where they managed to outperform DA for total travelled distance category although they did not have any bits of knowledge about the layout. From the findings as well, these algorithms shown a lack of significant different against the best performing algorithm for all categories considered which shown their competitiveness even though they did not have any information about the layouts.

CHAPTER 7

MOBILE ROBOT NAVIGATION PROBLEM (SYMMETRIC LAYOUT)

7.1 Introduction

This experiment is using the exact parameter setup and platform as in the previous chapter. The algorithms considered for this assessment are Fix PSO (Fix PSO), Linear Decreasing Inertia Weight PSO (Linear PSO), Constricted PSO (CPSO), Area Extended PSO (AEPSO), Constricted Area Extended PSO (CAEPSO), Dynamic Acceleration Coefficients PSO (DACPSO), and Dynamic Parameterisation PSO (DPPSO). For this experiment, Robot Operating System (ROS) is used as the platform to program the source code for all methods and embedded into a mobile robot. The details of ROS is discussed in the following sub-section.

Seven factors are considered to assess the performance of the selected algorithms in these experiments. The first factor considered is *Time* which the total time is taken for the robot to travel from the starting location to the goal location. The next factor considered is the *Number of Collisions* where the number of collisions occurred during the runs counted. *Arrived at Destination* where the ability of the algorithm to successfully drive the robot to the goal location depending on odometry are tested. *Travelled Distance* is one of the crucial factor considered where the total distance that the robot travelled from the starting location to the goal position is evaluated. Another important factor is *Battery Consumption*, where the total of percentage of battery consumed on average by the robot in order to complete the task assigned. *Displacement*

Problem is also measured as one of the factors where if the robot's final position is out of the tolerance range of the goal location, then it is considered as a displacement problem. The tolerance range is one and a half size of the mobile robot used which is 90cm radius from the centre of exact coordinate location.

7.2 The Symmetric Layout

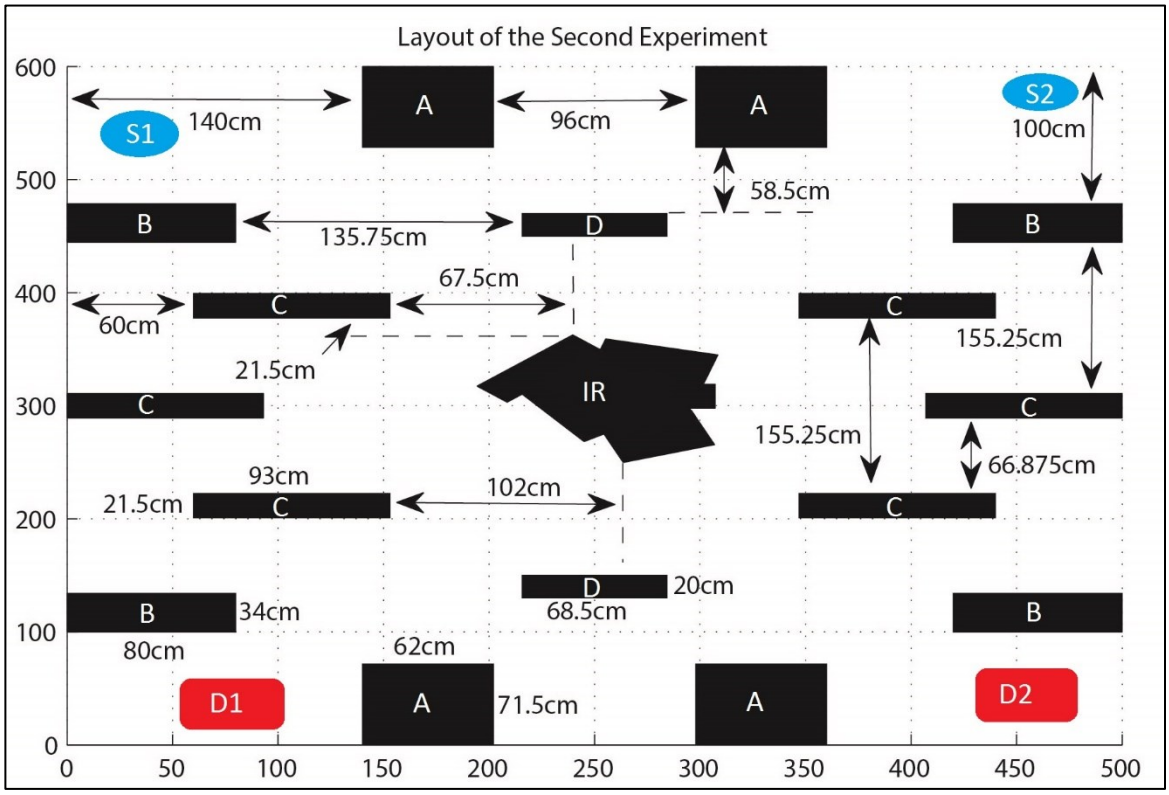


Figure 7-1. Symmetric Layout with an Irregular Shape Obstacle.

Figure 7-1 illustrates the layout of the third experiment. The layout is designed to be symmetric on both sides with irregular obstacles being placed in the middle. The environment is 5 meters wide by 6 meters long. This layout comprises of a combination of 4 sets of barriers

(differing in their dimensions) and one irregular-shaped obstacle. The dimensions (length (l) × width (w)) of the barriers utilised in this layout can be found in Table 7-1. Two starting points (marked as S1 and S2) and two destination points (marked as D1 and D2) are considered in this experiment. The results in three sub-experiments are different from each other as their starting and destination locations are altered.

Figure 7-2. The view of Path Planning Experiment (Symmetric Layout) from Four Difference Angles.

These sub-experiments include the following routes i) S1 to D1, ii) S2 to D1 and iii) S2 to D2. The sub-experiment that features the route between S2 to D1 is ignored due to the

existing symmetry in this layout. Figure 7-2 offers four snapshots from different angles of this symmetric layout.

Table 7-1. The dimensions of the utilised obstacles in the environment for symmetric layout for path planning experiment

Obstacle ID	Dimension
A	71.5*cm* × 62.0*cm*
B	34.0*cm* × 80.0*cm*
C	21.5*cm* × 93.0*cm*
D	68.5*cm* × 20.0*cm*
IR	552.5*cm* (parameter)

7.3 Result for Morphology Particle Swarm Optimisation

All performance from mobile robot navigation problems for symmetric layout is reported within four folds of three sub-experiments and overall achievements in Table 7-2, Table 7-3, Table 7-4, and Table 7-5. The results are discussed based on same categories from the previous layout which are i) number of obstacle collisions, ii) number of runs with successful arrival to the destination point, iii) number of runs with displacement problem, iv) average convergence iteration (for EA-based methods), v) average travel distance, vi) average battery consumption and vii) average execution time (s).

From the observations in sub-experiment 1 (Table 7-2), all algorithm shared the best performance in the first, second, and third factors considered (arrived at the destination, the number of collisions, and displacement problem) because all the algorithms used the same approach for obstacle avoidance and the same platform. For the time execution category, DA

needed the least amount of time to complete the task on average with the result of 87.5493 seconds. MPSO and CPSO came in second and third with the output of 124.8705 seconds and 126.6746 seconds on average respectively to complete the task.

In battery consumption category, DA once again managed to outperform other algorithms with a mean result of 0.3524% power consumption. CPSO is the second best performing algorithm with a mean result of 0.4637% energy consumption and followed by PF with an average outcome of 0.4970% power consumption.

Table 7-2. MPSO results for Path Planning in Symmetric Layout (Sub-Experiment 1)

Sub-Experiment 1							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	126.7588	0.4970	7.5604	-
DA	**10**	**0**	**0**	**87.5493**	**0.3524**	7.4706	-
RRT	**10**	**0**	**0**	385.1796	2.5631	9.2181	-
PRM	**10**	**0**	**0**	243.1524	0.9421	9.2613	-
DE	**10**	**0**	**0**	129.8887	0.7864	7.4456	470.6
GA	**10**	**0**	**0**	143.1296	0.8531	7.5950	519.6
CSA	**10**	**0**	**0**	178.1093	0.6565	7.5327	472.0
Fix PSO	**10**	**0**	**0**	150.0594	0.8939	7.7145	432.8
Rand PSO	**10**	**0**	**0**	145.7491	0.8605	7.6536	422.9
TVAC PSO	**10**	**0**	**0**	208.9403	1.2389	8.4111	422.7
Linear PSO	**10**	**0**	**0**	144.3843	0.8568	7.6117	449.1
CPSO	**10**	**0**	**0**	126.6746	0.4637	7.4822	476.5
MPSO	**10**	**0**	**0**	124.8705	0.5082	**7.3993**	**422.3**
CMPSO	**10**	**0**	**0**	127.9800	0.5604	7.4793	466.0

Surprisingly, MPSO managed to outperform DA in total travelled distance category with an average of 0.0713 meters short from DA's outcome. DE also managed to outclass DA in this category and become the second best performing algorithm with a mean result of 7.4456 meters. It is due to the risk is taken by MPSO and DE where they travelled nearer to the edges of the obstacles which reduced the total travelled cost as a result compares to DA. MPSO also outperformed others in the average convergence iteration category with 422.3 iterations. TVAC PSO closely follows this performance with 422.7 iterations and Rand PSO with 422.9 iterations.

Figure 7-3 illustrates the trajectory traces for employed approaches in Sub-Experiment 1 from experiment III. Each sub-figure in Figure 7-3 represent ten paths travelled by the turtlebot robot using a particular motion planning and navigation technique. These ten routes in each sub-figure are illustrated using different colour coding.

From the observation, almost all methods show fairly consistent performance with two different patterns of trajectory traces. PF and DA share the same trend (several turns) while the rest show the different type of path with almost no sharp turn at all but only one arch turn. This arc turn is near to the irregular obstacles placed in the middle of the layout. It is noteworthy that MPSO forms a trajectory traces like almost a straight line. It can be a reason why the overall travelled distance is the lowest compared to the other algorithms. RRT and PRM show less consistency and low precision in their trajectory traces as obviously seen in several diversions from the shortest path across ten executions. DA is expected to show the high consistency and precision in its trajectory traces since it has full knowledge about the layout. However, MPSO and DE also managed to perform surprisingly well as DA although they do not have the same prior knowledge as DA.

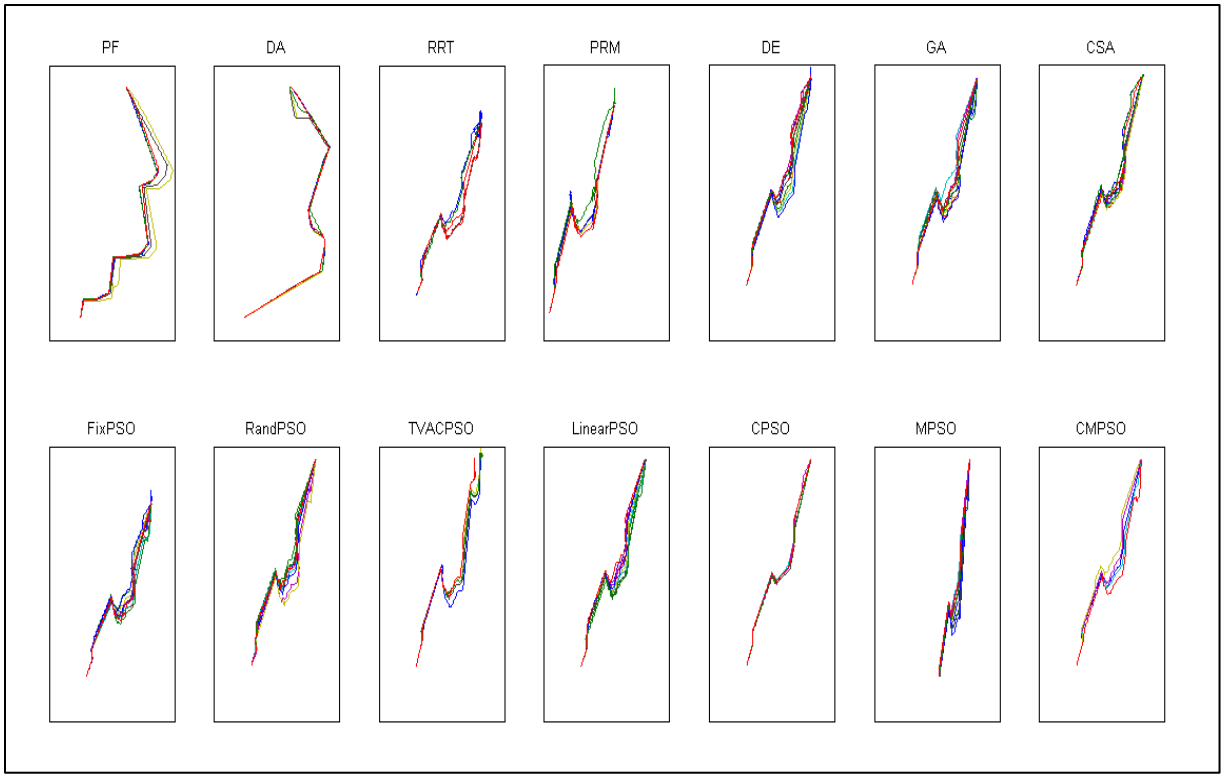


Figure 7-3. Trajectory traces of employed approaches (including Morphology based PSO) in Path Planning Experiment for Symmetric Layout (Sub-Experiment 1). Colour variation between trajectories is indicative of different executions (trials) with maximum ten trials.

In general, all algorithms seem to be using the same route or path to complete this task. It is caused by the irregular shape obstacle being placed in the middle of the environment which is likely to affect the algorithms chosen a path in a consistent way across all executions. Similar to the previous experiment, the layout of this experiment is divided into nine equal-sized rectangles named as regions. In sub-experiment 1, the starting and end points are located in regions 1 and 9 respectively with the irregular shape obstacle being placed in region 5 with partial coverage of region 8.

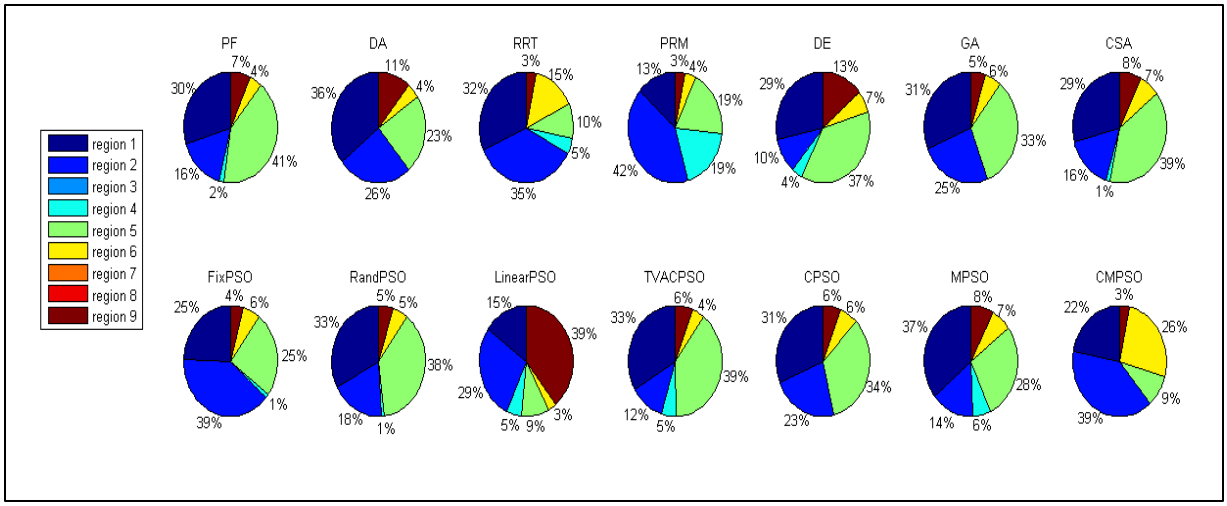


Figure 7-4. Pie Chart of Region Occupied for MPSO and CMPSO against other algorithms in Sub-Experiment 1 (Symmetric Layout).

The average number of the robot location in each region is reported in the format of pie chart in Figure 7-4 (across ten executions). From the observation on the pie chart, DA spent (on average) 36% of its time within region 1, 26% within region 2, 23% within region 5, 4% within region 6 and 11% within region 9. Meanwhile, PF spent 30% of its time within region 1, 16% within region 2, a small percent within region 4 (2%), 41% within region 5, 4% within region 6, and 7% within region 9. Linear PSO used most of its time in region 9 compares to the other algorithms with 39%. This might be due to the robot's inability to find the exact location of the final destination resulting in several passages of the same route between already visited points in this region. MPSO considered as the overall best performing algorithm in term of total travelled distance, considered region 5 as well in its path which occupied by irregular shape obstacles. As stated before, the consideration of region 5 (closer to obstacles) made the path shorter for MPSO hence outperform DA's outcomes for travelled distance category.

In this sub-experiment 2 (Table 7-3), DA outperformed all other motion planning methods in all categories except for iteration numbers which only competed amongst EA-based methods. Similar to previous experiments, no collision or displacement problems are observed in any of the executions of the approaches utilised. In the average execution time category, DA performance of 77.0312 seconds is followed by PF and MPSO with 106.6591 and 115.0737 seconds respectively. DA outperforms other algorithms in average battery consumption factor with 0.3115% followed by CPSO with 0.3131% and MPSO with 0. 3636% average battery consumption. DA is the only method managed to record an average travelled distance below 6 meters with the mean result of 5.9697 meters. MPSO is slightly higher with the mean of 6.1354 meters travelled distance. CPSO become the third best performing algorithm for total travelled distance category with 0.0599 meters more than MPSO and 0.2256 meters more than DA on average. Unexpectedly, Rand PSO required the least average iteration numbers to complete this task with the result of 401.5 iterations. Linear PSO becomes the second best with the result of 411.5 iterations and followed by CPSO with 470.0 iterations.

Table 7-3. MPSO results for Path Planning in Symmetric Layout (Sub-Experiment 2)

Sub-Experiment 2							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	106.6591	0.4154	6.2051	-
DA	**10**	**0**	**0**	**77.0312**	**0.3115**	**5.9697**	-
RRT	**10**	**0**	**0**	334.8004	1.2537	6.9449	-
PRM	**10**	**0**	**0**	334.5404	1.2537	6.9433	-
DE	**10**	**0**	**0**	118.0599	2.0363	6.5982	488.8
GA	**10**	**0**	**0**	154.5190	0.5786	6.9026	543.2
CSA	**10**	**0**	**0**	210.0471	0.7826	7.0973	652.2
Fix PSO	**10**	**0**	**0**	124.4046	3.8131	6.4567	497.8
Rand PSO	**10**	**0**	**0**	126.8047	0.4970	6.5045	**401.5**
TVAC PSO	**10**	**0**	**0**	130.4547	0.5119	6.5692	496.8
Linear PSO	**10**	**0**	**0**	123.4652	0.4748	6.5190	411.5
CPSO	**10**	**0**	**0**	115.9784	0.3131	6.1953	470.0
MPSO	**10**	**0**	**0**	115.0737	0.3636	6.1354	480.8
CMPSO	**10**	**0**	**0**	116.9857	0.3999	6.2150	474.9

The trajectory traces of the methods utilised in this experiment are illustrated in Figure 7-5. The results indicate that DA undertook the same path in all ten runs within this sub-experiment while GA, CSA, and RRT show some inconsistency in their travelled paths. PF, DE, and MPSO trajectory traces are almost close to DA trajectory traces. Between the EA-based approaches, DE choose paths which are closer to the ones chosen by DA with CSA demonstrating the most divergent from such path compared to the others. RRT used a different route once which visibly seen in the figure. Although, MPSO trajectory traces are not identical between one to another but the consistency in shape can be seen with majority of the routes taken are closer to the obstacles. Rand PSO, TVAC PSO, and Linear PSO showed the same behaviour in term of selecting the path with evidence can be seen from almost same trajectory traces.

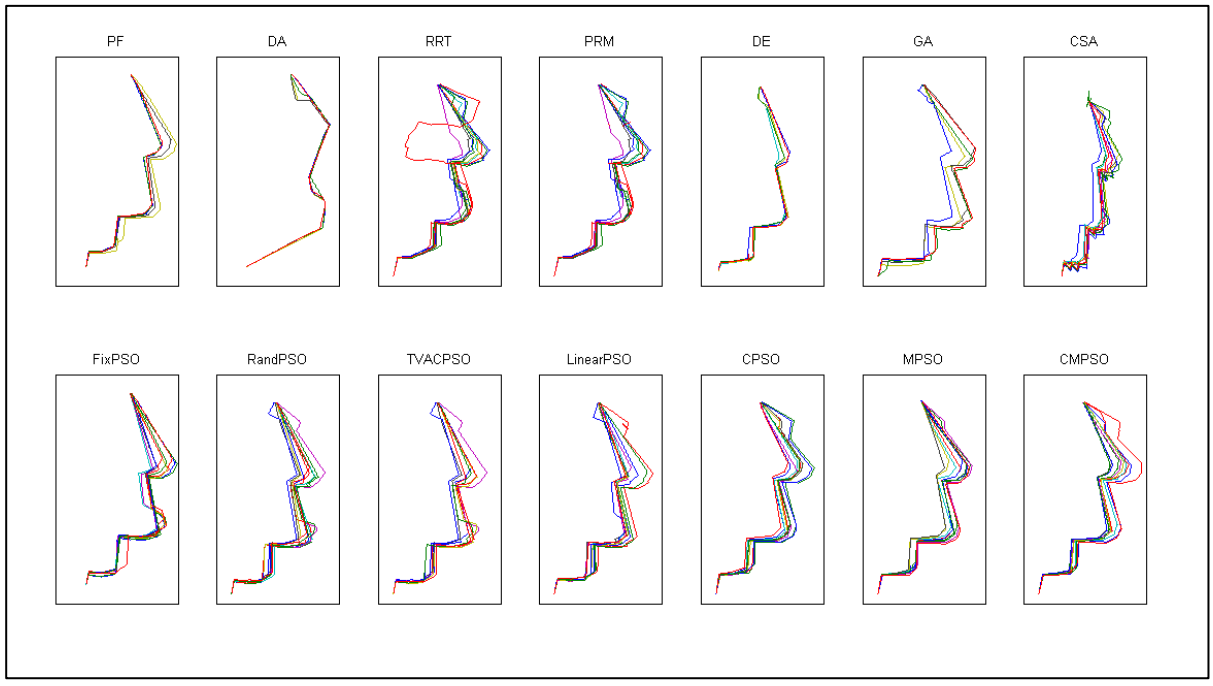


Figure 7-5. Trajectory traces of employed approaches (including Morphology based PSO) in Path Planning Experiment for Symmetric Layout (Sub-Experiment 2). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.

Result presented in Figure 7-6 indicate that all methods are consistent in their chosen trajectory by only visiting regions 1, 4, and 7 with an exception to CMPSO which visiting region 5 as well. The target destination is within region 7 while the starting point is within region 1. Looking at pie chart results in Figure 7-6, PF, DA, CPSO, and MPSO reported almost identical distribution across the three regions. The noticeable difference between the performances of these four algorithms is that DA has 3 to 5% higher average occupation of destination region while having around 1 to 4% less average occupation of region 1 compared to CPSO, MPSO, and PF. RRT and PRM demonstrated same pie chart performance with 40%, 39% and 21% average time spent in regions 1, 4 and 7 respectively. This similarity is also reflected in their trajectory traces presented in Figure 7-5 and similarities observed on various factors and categories reported in Table 7-3.

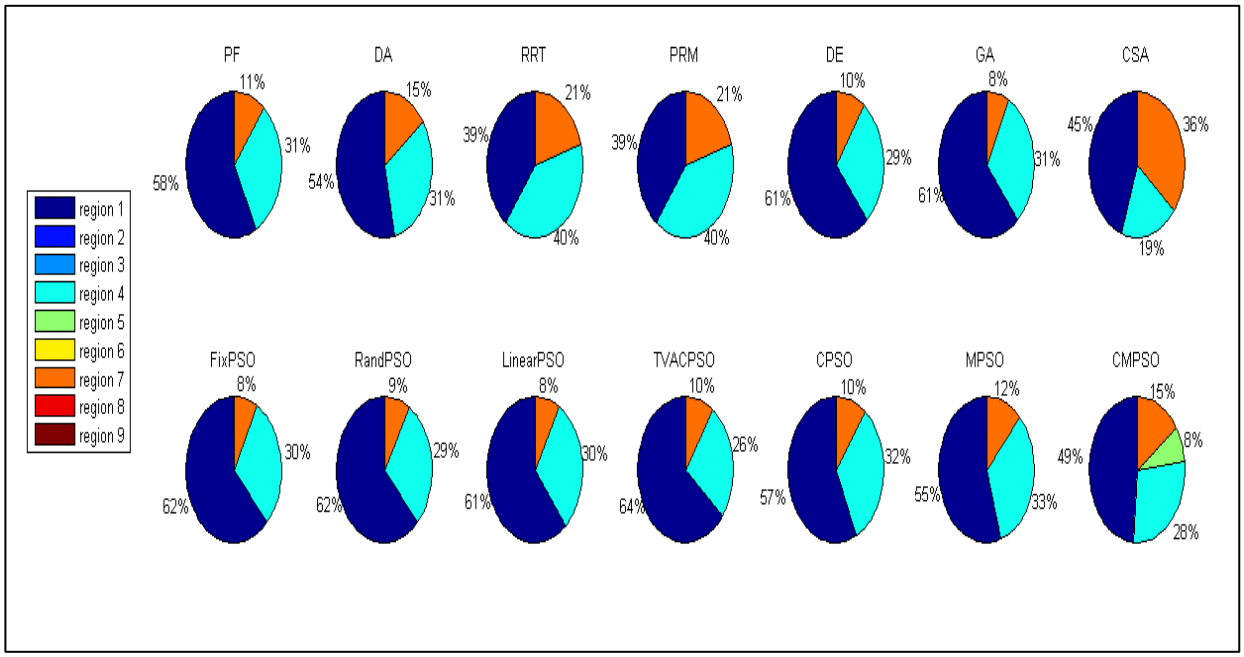


Figure 7-6. Pie Chart of Region Occupied for MPSO and CMPSO against other algorithms in Sub-Experiment 2 (Symmetric Layout).

GA and DE also reported a similar average percentage of coverage distribution across the three regions. There are also two noticeable reading in this figure where CMPSO is the only algorithm occupied region 5 (8% occupation on average), and only CSA occupied region 7 longer than any other algorithms with 36% on average. This issue for CSA better explains the poor performance of CSA reported in Table 7-3 in which in almost all categories it is identified as the worst performing approach.

Similar to previous sub-experiments, in this sub experiment, all algorithms managed to drive the robot to its destination safely without any obstacle collision and displacement problems as shown in Table 7-4. DA clocked up 90.6391 seconds in average for execution time category followed by MPSO (171.4892s) and CPSO (169.8085s). DA outperform other algorithms in battery consumption category with only 0.3709% average battery consumption compared with the second and third best achieved performances of 0.6565% (MPSO) and 0.7031% (CPSO). DA travelled the shortest path on average with 7.1247 meters followed by MPSO and CPSO with 7.9212 and 7.9538 meters respectively. CMPSO surprisingly outperforms other algorithms and become the best performing algorithm for iteration numbers category with average of 440.1 iterations. Rand PSO and MPSO become as second best and third best with an average iteration numbers of 444.2 and 445.4 respectively. The least performing algorithm is RRT for execution time and battery consumption with 374.3786 seconds and 1.3910% on average respectively. Meanwhile, for travelled distance category, GA recorded the highest travelled distance of 9.6089 meters on average, and for iteration numbers category, Linear PSO recorded 517.9 iteration numbers on average.

Table 7-4. MPSO results for Path Planning in Symmetric Layout (Sub-Experiment 3)

Sub-Experiment 3							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	184.3313	0.7270	8.4680	-
DA	**10**	**0**	**0**	**90.6391**	**0.3709**	**7.1247**	-
RRT	**10**	**0**	**0**	374.3786	1.3910	8.6183	-
PRM	**10**	**0**	**0**	209.6193	0.7937	8.0842	-
DE	**10**	**0**	**0**	192.5292	1.4948	8.7962	462.0
GA	**10**	**0**	**0**	252.7201	0.9830	9.6089	468.9
CSA	**10**	**0**	**0**	335.1722	1.3168	9.4214	723.5
Fix PSO	**10**	**0**	**0**	286.9791	1.1053	8.5221	502.9
Rand PSO	**10**	**0**	**0**	190.9152	0.7419	8.3127	444.2
TVAC PSO	**10**	**0**	**0**	215.9695	0.8309	8.9183	466.3
Linear PSO	**10**	**0**	**0**	299.9560	1.1573	8.7546	517.9
CPSO	**10**	**0**	**0**	179.8085	0.7031	7.9538	453.9
MPSO	**10**	**0**	**0**	171.4892	0.6565	7.9212	445.4
CMPSO	**10**	**0**	**0**	186.0757	0.8753	8.0144	**440.1**

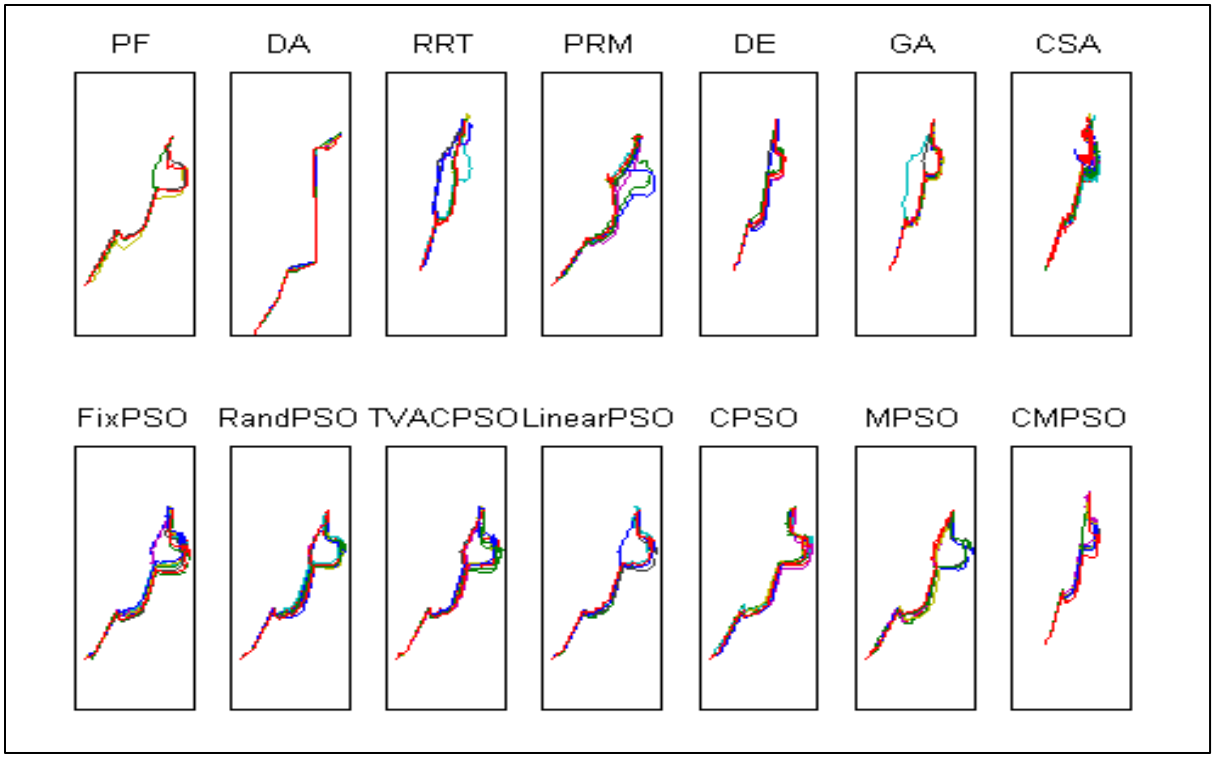


Figure 7-7. Trajectory traces of employed approaches (including Morphology based PSO) in Path Planning Experiment for Symmetric Layout (Sub-Experiment 3). Colour variation between trajectories is indicative of different executions (trials) with maximum ten trials.

The trajectory traces of the methods utilised in this experiment are illustrated in Figure 7-7Figure 7-7. Similar to the previous experiments; DA is consistent with its chosen path. CPSO also demonstrated consistency in its chosen path although it is not identical to DA. MPSO used a route which required turtlebot to go around the obstacles like path selected by CPSO twice. It is because of MPSO slight advantage over CPSO and made the total travelled distance shorter as evidence in Table 7-4. CSA and RRT showed a wandering behaviour near the goal point indicating their inability to identify the better path towards the goal. PRM showed the most similar trajectory to DA with only two executions following a slightly longer path. This is the reason why the total travelled distance of PRM is quite outstanding compared to its previous

results in travelled distance category. RRT and GA trajectory traces indicate inconsistent behaviours concerning the chosen manoeuvring strategies when they faced the irregular shape obstacle. CSA has shown quite a consistency at the early stage of manoeuvring, however, start to show inconsistency towards the later stage of manoeuvring which resulting in the weak outcome in the end.

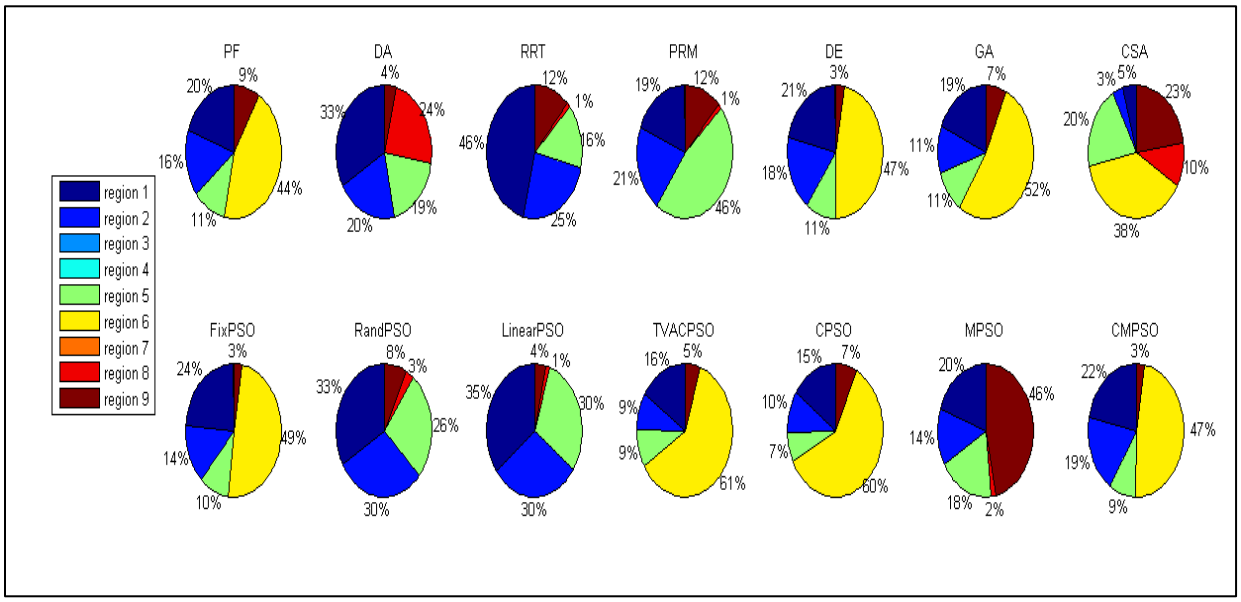


Figure 7-8. Pie Chart of Region Occupied for MPSO and CMPSO against other algorithms in Sub-Experiment 3 (Symmetric Layout).

For this sub-experiment 3, the starting point is within region 1 and the end point is within region 9. The results indicate that half of the algorithms inhabited region 6 as part of their path towards the destination while the other half of the algorithms did not consider this area at all. Considering DA as the best performing approach, it is noteworthy that region 6 is ignored while the highest average occupation percentage for region 8 is observed. DA occupied region 1 by 33% on average where only RRT, Rand PSO, and Linear PSO have reported higher or on par of average occupation percentage for this area. PRM and RRT demonstrated different region

coverages, where RRT occupied region 1 longer compared to PRM while PRM occupied region 5 longer than all other techniques. Unlike DA, RRT and PRM who excluded region 6 in their trajectories, CPSO recorded the second largest average occupation percentage for this area. The distribution of percentages between various parts in CPSO pie chart indicates that the robot controlled by CPSO performed exceptionally well in all regions but 6. It is likely due to the robot experiencing Cul-De-Sac problem by being surrounded by several obstacles and not being able to find a clear way out resulting in achieving poor performances in this region. It is noticeable that the average travelled distance differences between DA-CPSO and PRM-CPSO is in order of 0.5 and 0.03 meters respectively (see Table 7-4). This matter indicates that although CPSO performed poorly in region 6 but its exceptionally well performance in other regions (especially region 5 that contained the irregularly shaped obstacle) resulted in the algorithm becoming the third best performing method in this sub-experiment.

Although, MPSO did use path within region 6 twice, but since it just went through the region in short time hence, the percentage of region occupation is too small and neglected. MPSO is the only algorithm occupied region 9 longer than any other algorithms with 46%. However, this matter did not affect the final result of travelled distance achieved by MPSO. MPSO just used a friction of region 8 whilst completing this manoeuvring task. CMPSO occupied region 6 almost similar to MPSO's occupation on region 9 with 47% total occupation. CMPSO also required approximately even time to manoeuvre region 1 and region 2 with occupation percentage of 22% and 19% respectively.

Table 7-5. MPSO results for Path Planning in Symmetric Layout (Overall)

Overall								
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**	**Best Performing Algorithm**
PF	**10**	**0**	**0**	139.2497	0.5465	7.4111	-	9
DA	**10**	**0**	**0**	**85.0732**	**0.3449**	**6.8550**	-	17
RRT	**10**	**0**	**0**	364.7862	1.7359	8.2604	-	9
PRM	**10**	**0**	**0**	262.4374	0.9965	8.0963	-	9
DE	**10**	**0**	**0**	146.8259	1.4392	7.6133	473.8	9
GA	**10**	**0**	**0**	183.4562	0.8049	8.0355	510.6	9
CSA	**10**	**0**	**0**	241.1095	0.9186	8.0171	615.9	9
Fix PSO	**10**	**0**	**0**	187.1477	1.9374	7.5644	477.8	9
Rand PSO	**10**	**0**	**0**	154.4897	0.6998	7.4903	**422.9**	10
TVAC PSO	**10**	**0**	**0**	185.1215	0.8606	7.9662	461.9	9
Linear PSO	**10**	**0**	**0**	189.2685	0.8296	7.6284	459.5	9
CPSO	**10**	**0**	**0**	140.8205	0.4933	7.2104	466.8	9
MPSO	**10**	**0**	**0**	137.7818	0.5094	7.1520	449.5	11
CMPSO	**10**	**0**	**0**	143.0431	0.6119	7.2262	460.3	9

By considering the performances of all motion planning techniques across the three sub-experiments in this symmetric layout, DA has emerged as the best performing approach. DA show consistent top performance across all categories considered with only outperformed once by MPSO in the average travelled distance for sub-experiment 1. MPSO and CPSO are the second and third best performing algorithms in this experiment due to their consistently high performance in most categories across all three sub-experiments in this layout. PF shows quite promising results across all categories considered. Although, PF used local path planning approach, but since its concept of using highest force to determine the path chosen, it moves more directly towards goal and faster compared to RRT and PRM in local path planning.

Regarding comparison between classical and heuristic-based algorithms, it seems like heuristic-based methods performed decently especially in motion planning experiment for maze layout where all heuristic-based approaches managed to outperform all traditional algorithms (except DA) in all categories considered. However, in motion planning experiment for symmetric layout, PF managed to outperform all EA-based algorithms (except CPSO, MPSO, and CMPSO) in most of the categories considered. It is also worth mentioning that RRT and PRM performed quite poorly in terms of execution time and travelled distance. Although, these methods are actually among the best path planning algorithm, but they are only performing well if it is implements under global path planning condition where they have full or part knowledge about the environment layout.

Compared to another classical method, PF managed to outperform RRT and PRM in all categories. The reason is because of PF algorithm is more direct and did not have any influence on random factor compared to those two. Although DA managed to outperform other methods in all relevant categories, it is noteworthy that DA has an advantage compared to others. Unlike

the other methods that have zero knowledge about the environment layout forcing them to perform local motion planning base on their online sensory readings, DA had access to the complete environmental plan and uses global path planning. Therefore, the results achieved by DA is considered as the perfect optimal performance. From the overall performance, it can be concluded that MPSO is the best performing approach given that it became the second best to DA in almost every category considered in the research.

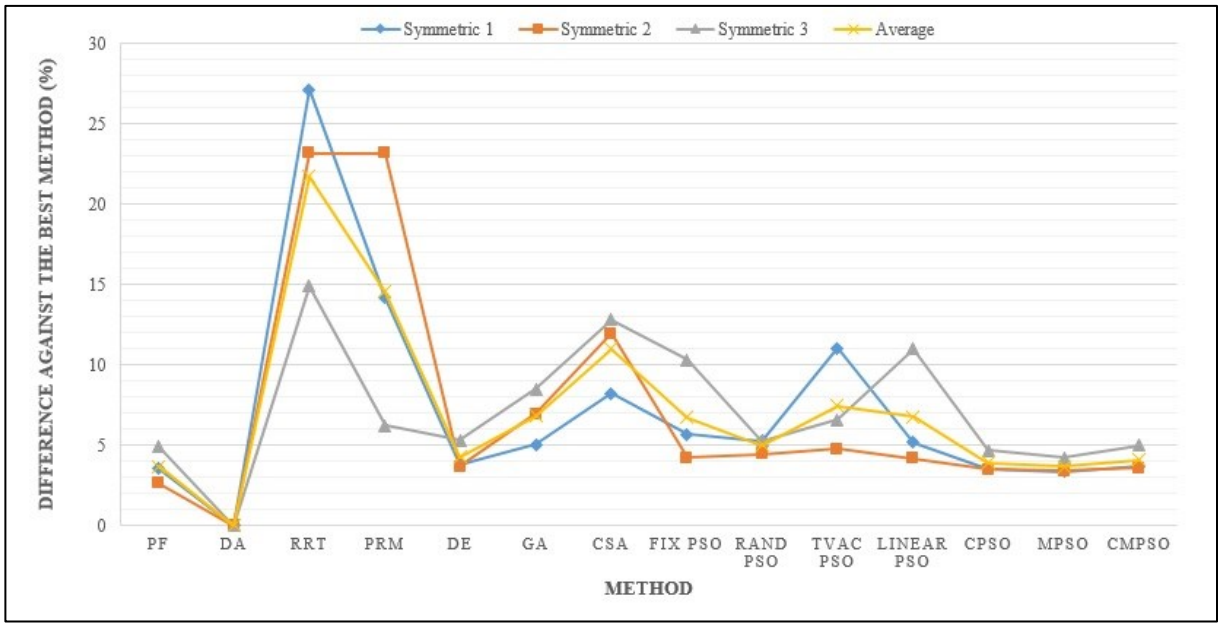


Figure 7-9. The graph of the best performing algorithm against other algorithms (Execution Time Factor).

Figure 7-9 illustrates the best performing algorithm as the benchmark for the other algorithms for execution time category. There are consistent performances shown across 3 paths chosen by PF, DE, Rand PSO, CPSO, MPSO, and CMPSO. However, in average, MPSO and PF are quite close to DA as the different between them against DA is less than five percent. The worst performance is seen in the first path where RRT recorded more than twenty-five percent difference against the DA.

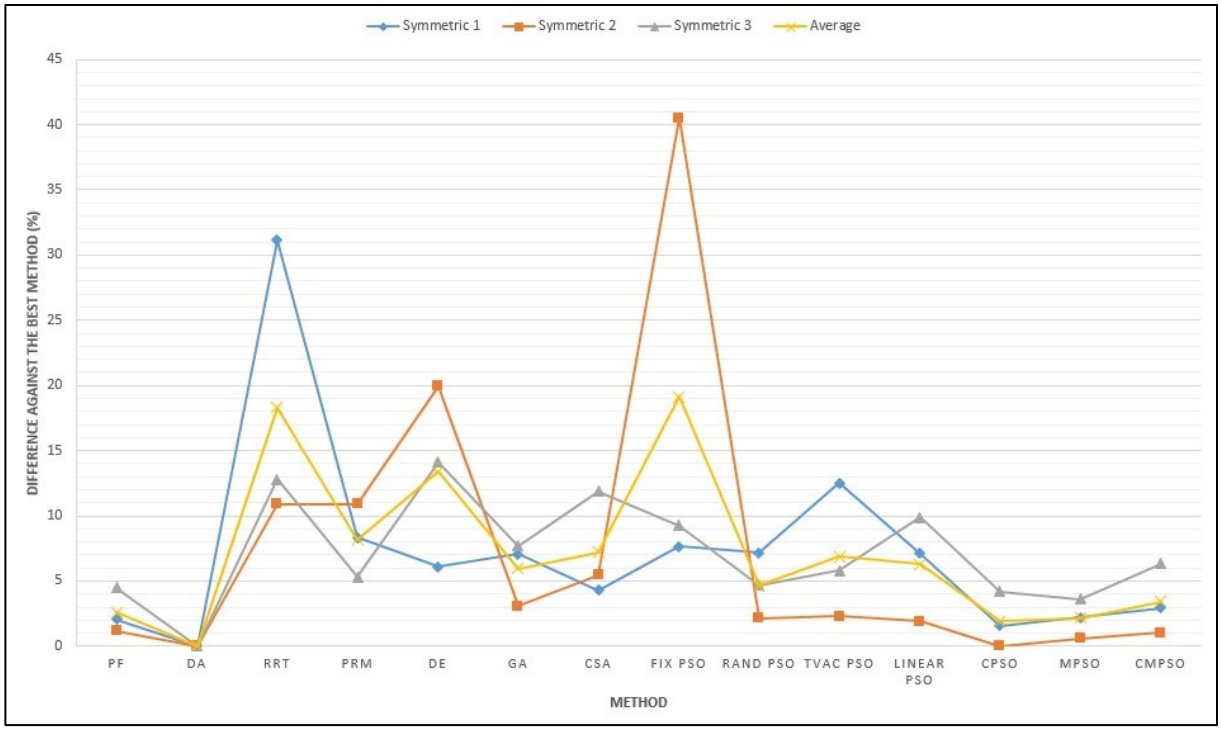


Figure 7-10. The graph of the best performing algorithm against other algorithms (Battery Consumption Factor).

The graph of the percentage difference between the best performing algorithm and the other algorithms for battery consumption category is illustrated in Figure 7-10. The most noticeable difference can be seen in RRT's performance in the first path and Fix PSO's performance in the second path with more than thirty percent and thirty-five different against DA's result respectively. The consistency can be seen in PF, CPSO, MPSO, and CMPSO as all of them shown similar performance amongst them for all paths. CPSO also recorded almost par result with the DA in the second experiment.

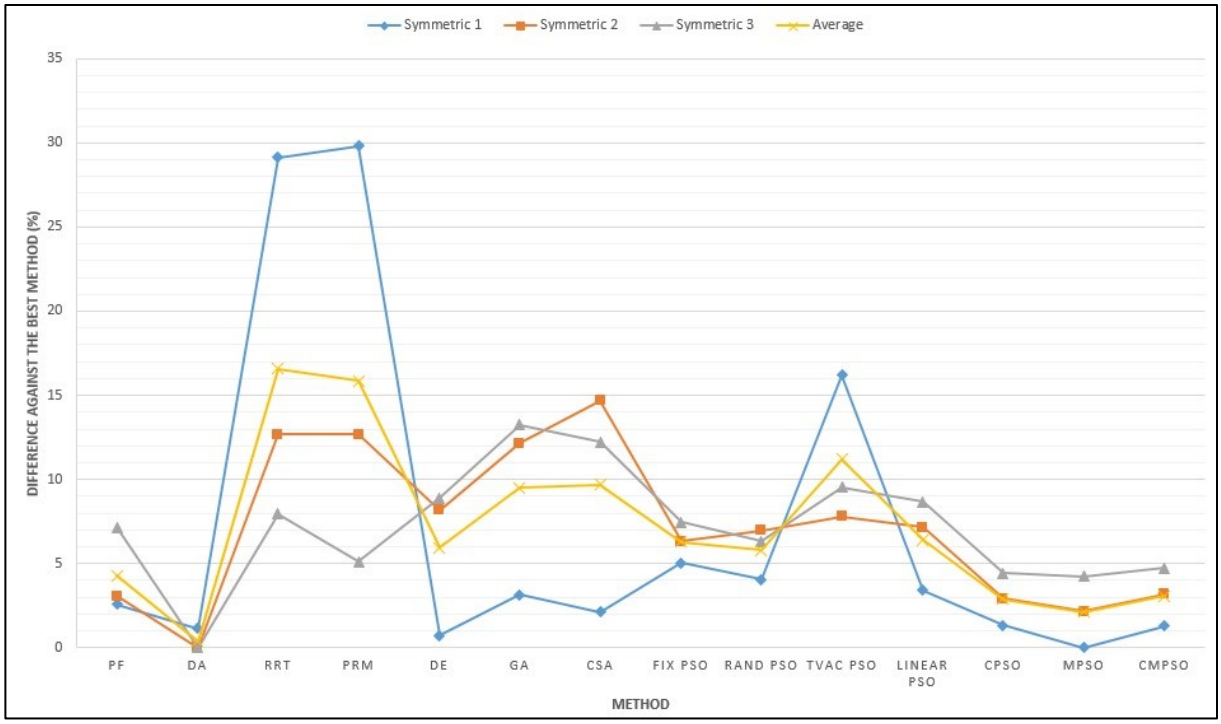


Figure 7-11. The graph of the best performing algorithm against other algorithms (Travelled Distance Factor).

MPSO has shown better performance than DA in the first path in term of travelled distance factor as illustrated in Figure 7-11. DE also managed to outperform DA in the first path, but the different is less than one percent. The worst performing algorithm is PRM which is nearly thirty percent different against MPSO. The algorithms which recorded the consistence performance beside DA are CPSO, MPSO, and CMPSO. Based on the average result, it shows only PF, CPSO, MPSO and CMPSO recorded less than five percent against DA. On the other hand, only RRT, PRM and TVAC PSO registered more than 10 percent difference against DA.

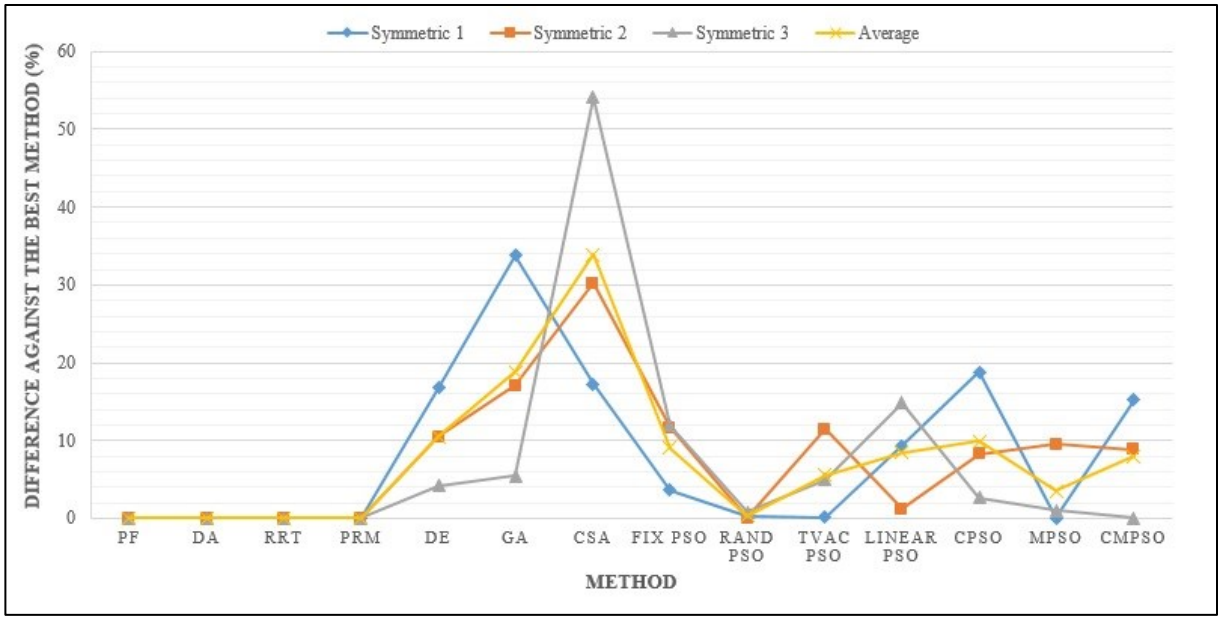


Figure 7-12. The graph of the best performing algorithm against other algorithms (Iterations Factor)

PF, DA, RRT and PRM are not considered in this category as they are not evolutionary algorithms. Rand PSO is the best performing algorithm for this category as shown in Figure 7-12. MSPO is the only algorithm which recorded less than ten percent different against Rand PSO for all three paths considered for this experiment. CSA recorded two biggest difference against RAND PSO in the second and third sub-experiments while GA recorded the most significant difference in the first sub-experiment against RAND PSO.

7.4 Significance Analysis for Morphology Particle Swarm Optimisation

Table 7-6. Significance Analysis for Morphology Particle Swarm Optimisation (MPSO).

Category	Experiments	Approaches	Experiments & Approaches
Time (s)	p = 2.90108e-27	p = 5.13176e-89	p = 2.10670e-27
Battery Consumption (%)	p = 7.22500e-01	p = 4.00000e-04	p = 4.16000e-02

Travelled Distance (m)	p = 2.79102e-58	p = 9.89340e-17	p = 2.98832e-28
Convergence iteration	p = 4.48000e-02	p = 0	p = 0

The same statistical significant analysis of performance are carried for these experiments involved. The significant of findings on the basis of categories and factors considered earlier are discussed below:

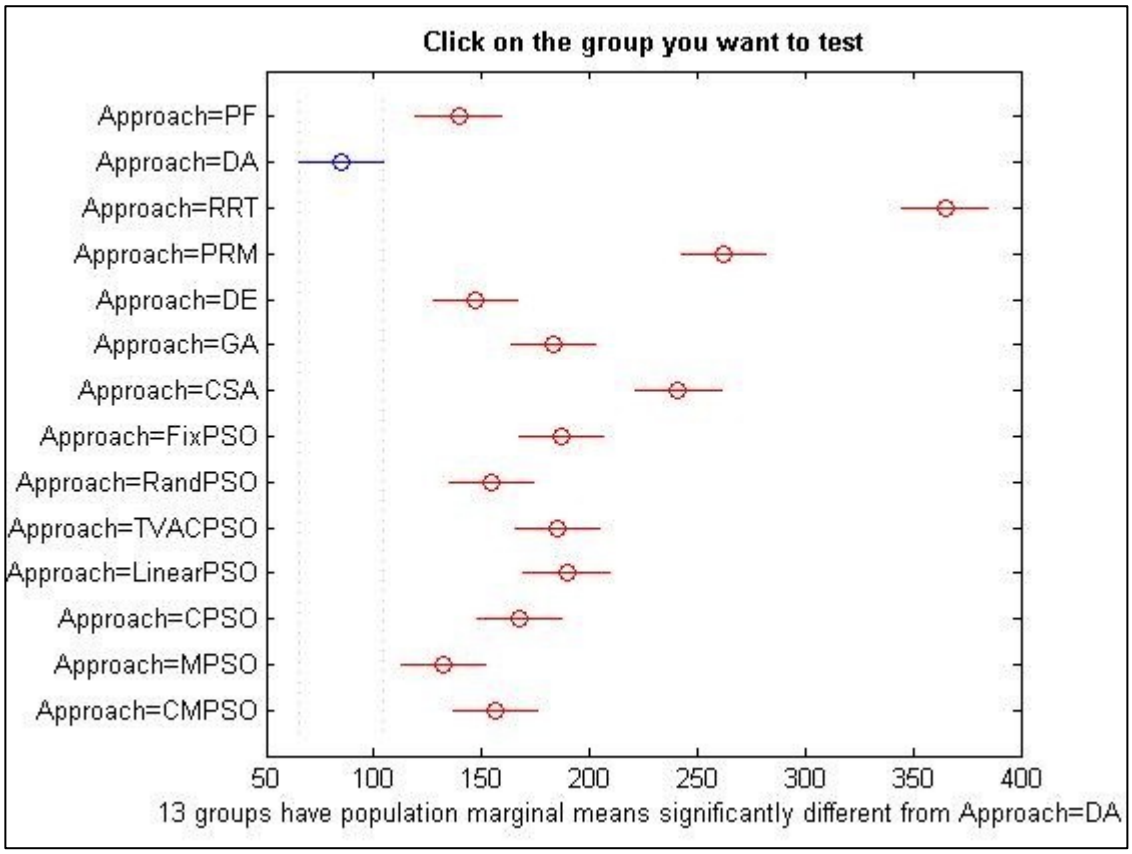


Figure 7-13. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Execution Time factor.

- Execution time (s): There is significant difference exists between approaches utilised in this category (p=5.13176e-89<0.05). DA shown statistical difference against all algorithms. However, considering the advantage the DA has, it means MPSO is the best

performing algorithm. MPSO shown significant difference between seven other methods which is RRT, PRM, GA, CSA, Fix PSO, Rand PSO, and TVAC PSO.

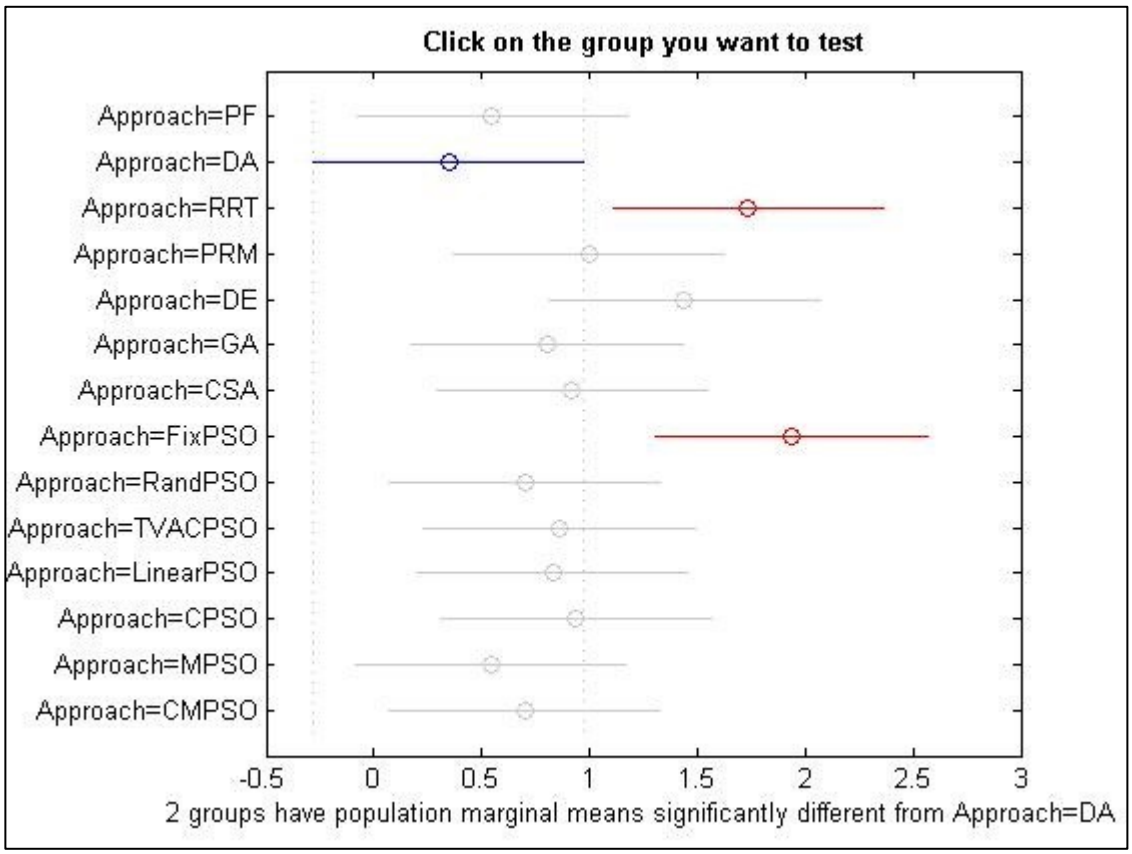


Figure 7-14. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Energy Consumption factor.

- Energy consumption (%): The results of the statistical significant analysis indicated such significant in approaches (p=0.0004<0.05). Within the algorithms applied, DA showed significant difference against RRT and Fix PSO with PF and MPSO showed significant difference against Fix PSO. Meanwhile, other algorithms are shown a lack of statistical difference amongst themselves.

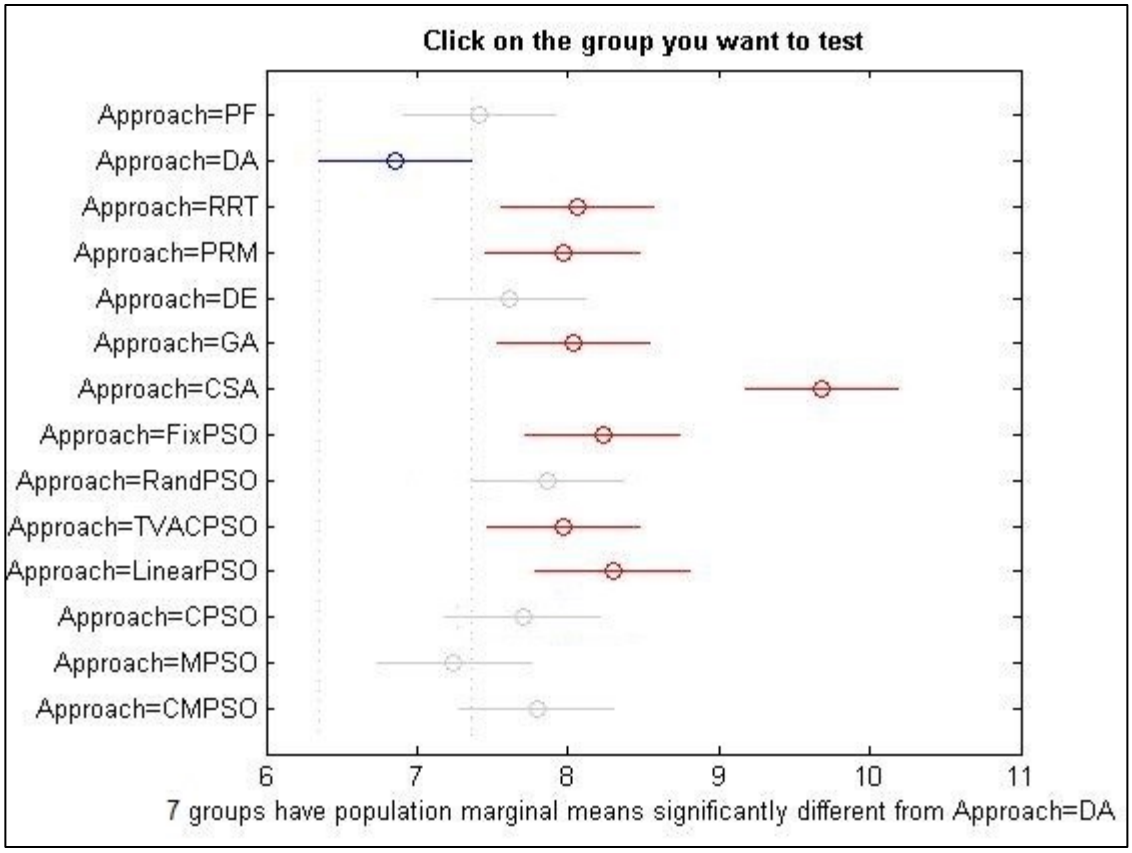


Figure 7-15. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Travelled Distance factor.

- The length of the travelled path (*m*): From the observation, the results shown the existence of a significant difference between approaches (p = 9.8934e-17 < 0.05). The result indicated a significant difference between DA against other methods employed. PF, DE, CPSO, MPSO, and CMPSO showed no significant difference between them but PF and MPSO show significant difference against the other algorithms. DE and CMPSO showed the existence of significant different between them against RRT, PRM, CSA, Fix PSO and Linear PSO.

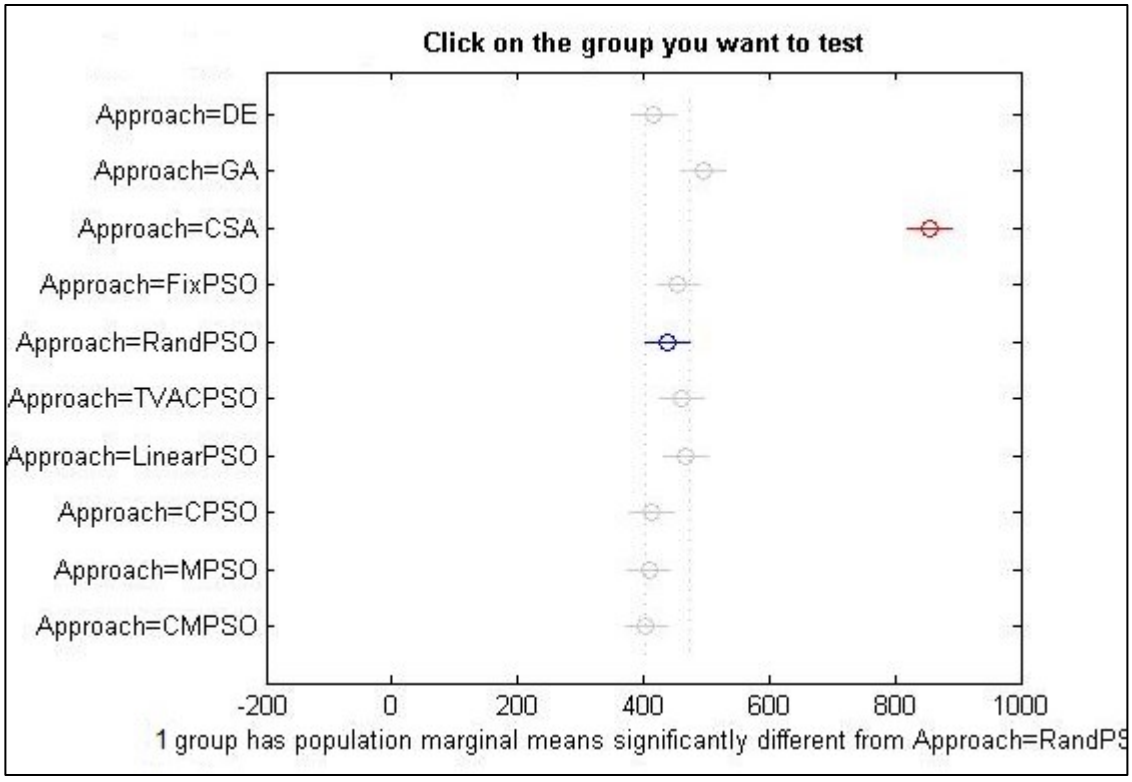


Figure 7-16. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Iteration factor.

- Convergence iteration: The statistical analysis of the results point out significant different between approaches (p=0.0000<0.05). GA, CSA and TVAC PSO shown significance different against other approaches including amongst themselves.

7.5 Result for Dynamic Approaches of Particle Swarm Optimisation

All performance from mobile robot navigation problems are reported in Table 7-7 (sub-experiment 1), Table 7-8 (sub-experiment 2), and Table 7-9 (sub-experiment 3). The results are discussed based on the same categories; i) number of obstacle collisions, ii) number of runs with a successful arrival to the destination point, iii) number of runs with a displacement problem, iv) the average of convergence iteration (for EA-based methods), v) the average of

travelled distance, vi) the average of battery consumption and vii) the average of execution time (s).

From the observations in sub-experiment1, the first factor considered is the number of collisions. The result is similar to the findings in the previous layout where all algorithms utilised managed to avoid collision while moving from the starting point towards the destination. All approaches also managed to arrive at the destination and without any displacement problem. DA outperformed other methods concerning the average execution time where it clocked only 87.5493 seconds to complete the navigation task. DPPSO and CAEPSO followed DA's performance in this category with 120.3214 seconds and 123.4300 seconds respectively. DA become the best performing algorithm in battery consumption factor with only using 0.3524% energy on average. DPPSO follows this performance with 0.4432% and CPSO with 0.4637% average power consumptions.

For travelled distance category, DE, CAEPSO, and DPPSO managed to outperform DA with an average travelled distance of 7.4456 meters, 7.4333 meters, and 7.3982 meters respectively. DA only come as fourth best performing algorithm with 7.4706 meters. It is perhaps because unlike DA, that places its nodes in the middle point of two nearby obstacles (a safe distance from either obstacle), those three methods are taking a risk by allowing the turtlebot manoeuvring closer to the edges of the obstacles which made the final travelled distance shorter.

TVAC PSO outperformed others in the average convergence iteration category with 422.7 iterations. Rand PSO closely follows this performance with 422.9 iterations and Fix PSO with 432.8 iterations.

Table 7-7. DAPSO Results for Path Planning in Symmetric Layout (Sub-Experiment 1)

Sub-Experiment 1							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	126.7588	0.4970	7.5604	-
DA	**10**	**0**	**0**	**87.5493**	**0.3524**	7.4706	-
RRT	**10**	**0**	**0**	385.1796	2.5631	9.2181	-
PRM	**10**	**0**	**0**	243.1524	0.9421	9.2613	-
DE	**10**	**0**	**0**	129.8887	0.7864	7.4456	470.6
GA	**10**	**0**	**0**	143.1296	0.8531	7.5950	519.6
CSA	**10**	**0**	**0**	178.1093	0.6565	7.5327	472.0
Fix PSO	**10**	**0**	**0**	150.0594	0.8939	7.7145	432.8
Rand PSO	**10**	**0**	**0**	145.7491	0.8605	7.6536	422.9
TVAC PSO	**10**	**0**	**0**	208.9403	1.2389	8.4111	**422.7**
Linear PSO	**10**	**0**	**0**	144.3843	0.8568	7.6117	449.1
CPSO	**10**	**0**	**0**	126.6746	0.4637	7.4822	476.5
AEPSO	**10**	**0**	**0**	133.2502	0.7231	7.5663	506.5
CAEPSO	**10**	**0**	**0**	123.4300	0.6385	7.4333	506.3
DACPSO	**10**	**0**	**0**	140.3164	0.5719	7.7772	504.3
DPPSO	**10**	**0**	**0**	120.3214	0.4432	**7.3982**	478.0

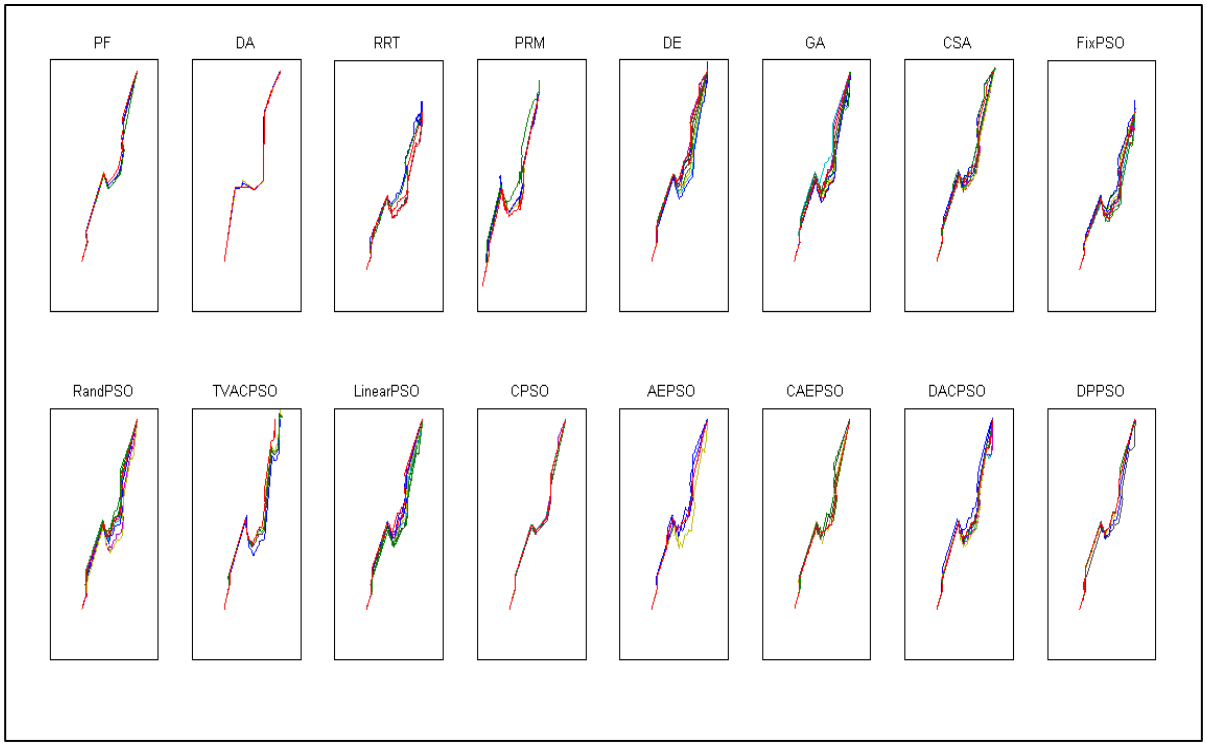


Figure 7-17. Trajectory traces of employed approaches in Path Planning Experiment for Symmetric Layout (Sub-Experiment 1). Colour variation between trajectories is indicative of different executions (trials) with maximum ten trials.

Figure 7-17 illustrates the trajectory traces for employed approaches in Sub-Experiment 1 from the symmetric layout. Each sub-figure in Figure 7-17 represent ten routes travelled by the turtlebot robot equipped with a precise motion planning and navigation techniques. All ten paths in each sub-figure are described using different colour coding. From the observation, PF, DA, CPSO, AEPSO, CAEPSO, and DPPSO show the most consistent performance with almost identical paths in ten attempts. RRT, DE, TVAC PSO, and Fix PSO show less consistency and low precision in their trajectory traces as seen in the tracks across ten runs. DA is expected to show coherence and high precision in its trajectory traces since it is required full knowledge of the layout to navigate. However, CAEPSO and DPPSO are surprisingly outclassed DA in term

of total travelled distance although they do not have the same prior knowledge as DA. It shows that CAEPSO and DPPSO probably using the path closer to the obstacles which made the total travelled distance shorter. All algorithms seem to like using the same route or path to complete this mission. It could be the result of the irregular shape obstacle being placed in the middle of the environment which is unlikely to affect the algorithms chosen a path in a consistent way across all executions.

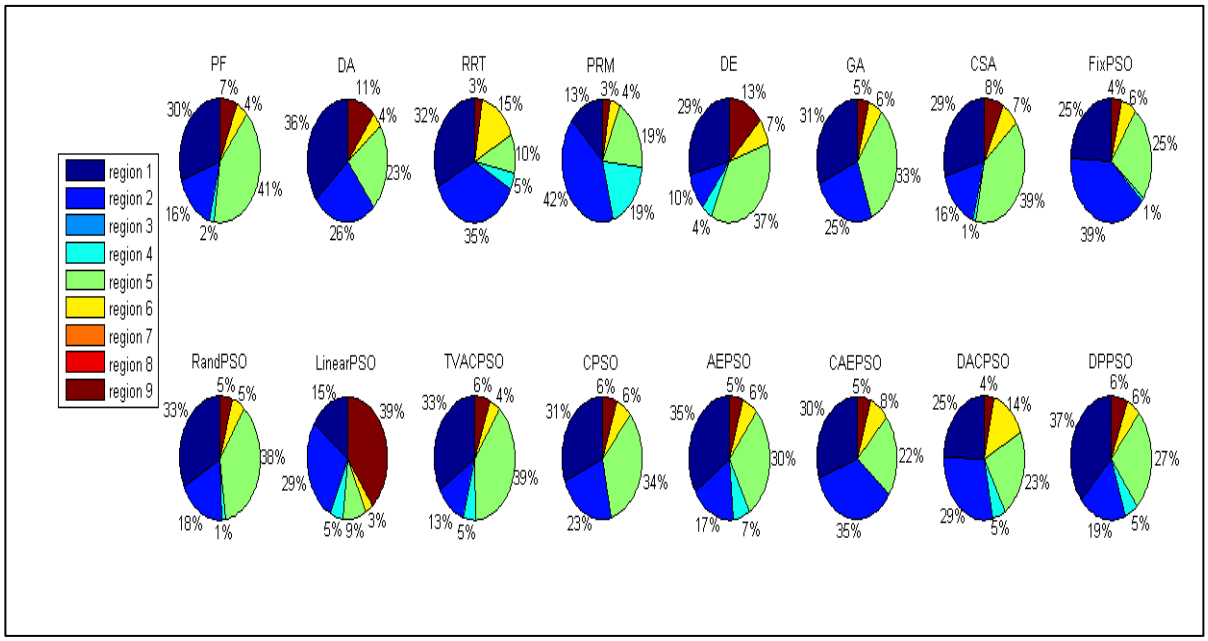


Figure 7-18. Pie Chart of Region Occupied for Dynamic Approaches PSO against other algorithms in Sub-Experiment 1 (Symmetric Layout).

Similar to previous navigation experiments, the layout of sub-experiment 1 for symmetric layout is divided into nine equal-sized rectangles named as regions. In sub-experiment 1, the starting and end points are located in areas 1 and 9 respectively with the irregular shape obstacle being placed in region 5 with partial coverage of region 8. The average

number of times (across ten runs) that the robot is located in each region is reported in the format of the pie chart in Figure 7-18.

DA, as the best performing method, spent (on average) 36% of its time within region 1, 26% within region 2, 23% within region 5, 4% within region 6 and 11% within region 9. GA, CPSO, and CAEPSO reported similar occupied areas as DA while the other algorithm considered region 4 as well. However, except PRM, others just occupied region 4 7% or less of their times. CAEPSO and DPPSO reported similar areas as DA with a significant difference in region 9 where CAEPSO and DPPSO manoeuvre slightly better with less 5-7% than DA. This small marginal differences between DA, CAEPSO, and DPPSO as observed in the pie charts are also supported by findings reported in Table 7-7 in which the average travelled distance differences between DA and CAEPSO is 0.373 meter and between DA and DPPSO is 0.724 meter. DACPSO reported similar region occupation as DPPSO as well but in term of percentages are slightly different. The major different are the possession in region 1, region 2, and region 6 where DPPSO out-manoeuvre DACPSO in region 2 and region 6 while DACPSO manoeuvre slightly efficient than DPPSO in region 1 with only 25%.

In sub-experiment 2 (Table 7-8), DA managed to outperform all other motion planning methods in all considered factor except for iteration numbers which only measured amongst EA-based methods. Similar to previous experiments, no collision or displacement problems are observed in any of the executions of the algorithms utilised. In the execution time factor, DA clocked 77.0312 to become the best performing and followed by DPPSO and CAEPSO as second and third best performing algorithm with 103.4310 and 104.1725 seconds respectively.

Table 7-8. DAPSO Results for Path Planning in Symmetric Layout (Sub-Experiment 2)

Sub-Experiment 2							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	106.6591	0.4154	6.2051	-
DA	**10**	**0**	**0**	**77.0312**	**0.3115**	**5.9697**	-
RRT	**10**	**0**	**0**	334.8004	1.2537	6.9449	-
PRM	**10**	**0**	**0**	334.5404	1.2537	6.9433	-
DE	**10**	**0**	**0**	118.0599	2.0363	6.5982	488.8
GA	**10**	**0**	**0**	154.5190	0.5786	6.9026	543.2
CSA	**10**	**0**	**0**	210.0471	0.7826	7.0973	652.2
Fix PSO	**10**	**0**	**0**	124.4046	3.8131	6.4567	497.8
Rand PSO	**10**	**0**	**0**	126.8047	0.4970	6.5045	**401.5**
TVAC PSO	**10**	**0**	**0**	130.4547	0.5119	6.5692	496.8
Linear PSO	**10**	**0**	**0**	123.4652	0.4748	6.5190	411.5
CPSO	**10**	**0**	**0**	115.9784	0.3131	6.1953	470.0
AEPSO	**10**	**0**	**0**	125.2043	0.4971	6.1237	462.6
CAEPSO	**10**	**0**	**0**	104.1725	0.4914	6.0162	441.7
DACPSO	**10**	**0**	**0**	126.2115	0.4711	6.3482	452.3
DPPSO	**10**	**0**	**0**	103.4310	0.4785	6.0241	451.5

In battery consumption factor, DA outperformed other algorithms with 0.3115% battery consumption on average. CPSO closely follows it with an average of 0.3131% and PF with an average 0.4154% on battery consumption. DA shown its superiority with becoming the only method recorded an average travelled distance under 6 meters (5.9697 meters). CAEPSO recorded just slightly higher average results with 6.0162 meter travelled distance. DPPSO recorded 0.0079 meters more than CAEPSO and 0.0544 meters more than DA on average. Rand PSO surprisingly becomes the algorithm which required the least average iteration numbers with the result of 401.5 iterations. Linear PSO and DPPSO become the second and third best with the result of 411.5 iterations and 451.5 iterations respectively.

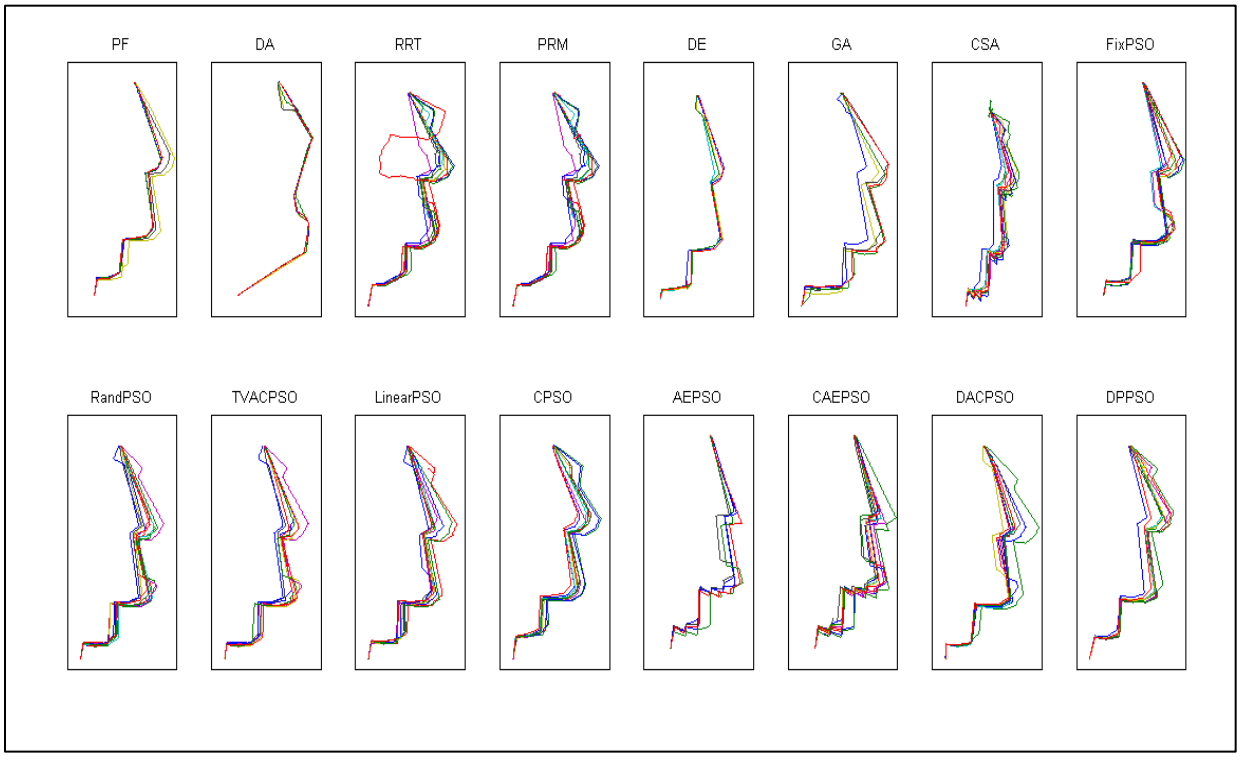


Figure 7-19. Trajectory traces of employed approaches in Path Planning Experiment for Symmetric Layout (Sub-Experiment 2). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.

The trajectory traces of the methods utilised in this experiment are illustrated in Figure 7-19. The results show clearly that PF, DA, and DE undertook the same path in all ten runs within this sub-experiment while RRT is the least consistent algorithm compared to the others in its travelled paths. Others algorithms were using almost similar path across their ten executions although they were not as precise as PF, DA, and DE. Although CAEPSO and DPPSO as not consistence as PF, DA, and DE but some of their paths are quite close to obstacles which made up to the other paths. It is also made their outcome for total travelled distance quite short. Between the EA-based methods, DE trajectories are close to the best performing algorithm (DA) with CSA demonstrating the most opposing from such path compared to the others even its trajectory traces are quite consistent.

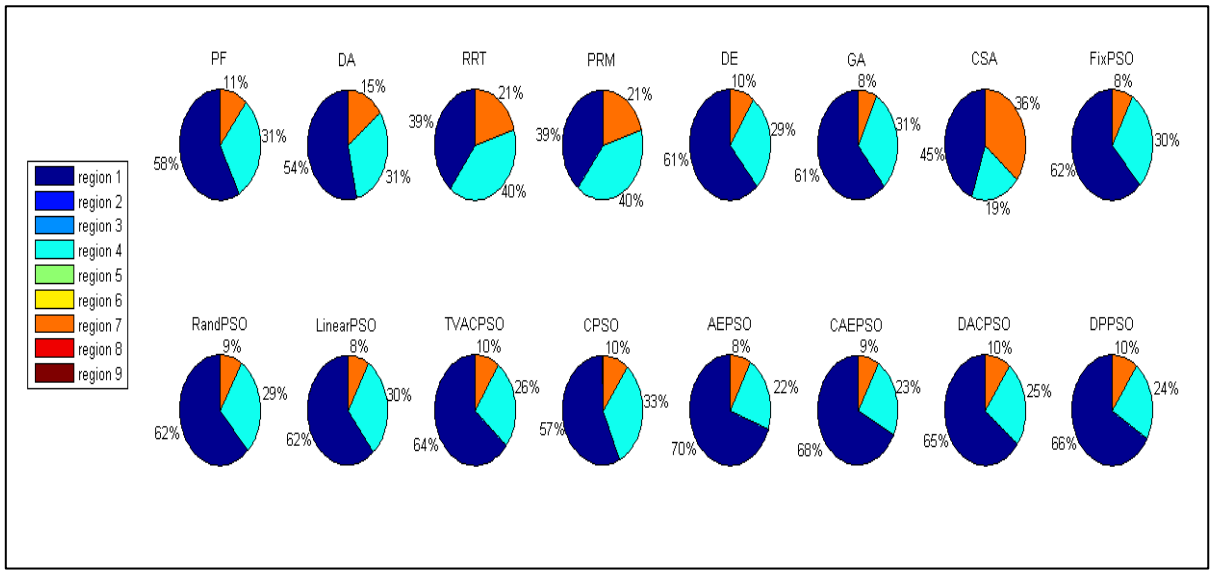


Figure 7-20. Pie Chart of Region Occupied for Dynamic Approaches PSO against other algorithms in Sub-Experiment 2 (Symmetric Layout).

Figure 7-20 shown pie chart representation of turtlebot's manoeuvres in different regions occupation in the experiment layout. In this sub-experiment 2, the initial position of

turtlebot is placed within region 1 and final positions is positioned in regions 7. Irregular shape obstacle is put in region 5 with partially within region 4.

Results presented in Figure 7-20 points out that all methods are consistent in their chosen trajectories by only visiting three regions across ten execution (region 1, region 4, and region 7). Focusing at the pie chart results in Figure 7-20, DA and PF reported almost same region occupation across the three areas. The noticeable difference between the performances of these three algorithms is that DA has 4% higher average occupation of destination region while having around 4% less average occupation of starting region compared PF. RRT and PRM demonstrated matching pie chart performance with 39%, 40% and 21% average region occupation in areas 1, 4 and 7 respectively. The three dynamic behaviours based PSO also demonstrated almost identical pie chart performance with just 1% occupation different amongst themselves for all three regions considered. The most different algorithm for this pie chart results is CSA with more time spent in region 7 compared to the others. This issue could be clarified the poor performance of CSA recorded in Table 7-8 where it is identified as the worst performing algorithm in almost all categories considered.

Similar to previous sub-experiments, in this sub-experiment 3, all algorithms managed to steer the turtlebot to its target securely by avoiding any obstacle collision and displacement problems as shown in Table 7-9. DA recorded 90.6391 seconds in average for execution time category and became the best performing algorithm. It is followed by DPPSO with a mean execution time of 156.0103 seconds and CAEPSO with a mean execution time of 157.4145 seconds. DA also outperform other algorithms in battery consumption factor with only 0.3709% average battery consumption compared with the second (0.5712% for CAEPSO) and the third (0.6237% for DPPSO) best performing algorithm.

Table 7-9. DAPSO Results for Path Planning in Symmetric Layout (Sub-Experiment 3)

Sub-Experiment 3							
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**
PF	**10**	**0**	**0**	184.3313	0.7270	8.4680	-
DA	**10**	**0**	**0**	**90.6391**	**0.3709**	**7.1247**	-
RRT	**10**	**0**	**0**	374.3786	1.3910	8.6183	-
PRM	**10**	**0**	**0**	209.6193	0.7937	8.0842	-
DE	**10**	**0**	**0**	192.5292	1.4948	8.7962	462.0
GA	**10**	**0**	**0**	252.7201	0.9830	9.6089	468.9
CSA	**10**	**0**	**0**	335.1722	1.3168	9.4214	723.5
Fix PSO	**10**	**0**	**0**	286.9791	1.1053	8.5221	502.9
Rand PSO	**10**	**0**	**0**	190.9152	0.7419	8.3127	444.2
TVAC PSO	**10**	**0**	**0**	215.9695	0.8309	8.9183	466.3
Linear PSO	**10**	**0**	**0**	299.9560	1.1573	8.7546	517.9
CPSO	**10**	**0**	**0**	179.8085	0.7031	7.9538	453.9
AEPSO	**10**	**0**	**0**	205.5403	0.9050	7.9706	451.5
CAEPSO	**10**	**0**	**0**	157.4145	0.5712	7.4107	443.3
DACPSO	**10**	**0**	**0**	201.9664	0.6528	7.8479	482.6
DPPSO	**10**	**0**	**0**	156.0103	0.6237	7.5545	**430.5**

DA managed to find the shortest path on average for total travelled distance with 7.1247 meters and closely followed by CAEPSO and DPPSO with 7.4107 meters and 7.5545 meters respectively. DPPSO managed to become the best performing algorithm for iteration numbers category with an average of 430.5 iterations per execution. CAEPSO and Rand PSO recorded average iteration numbers of 443.3 and 444.2 respectively to become the second best and the third best performing algorithm. RRT is the least performing algorithm is for execution time and battery consumption with 374.3786 seconds and 1.3910% on average respectively while for travelled distance factor, GA recorded the highest travelled distance of 9.6089 meters on average. Linear PSO recorded 517.9 iteration numbers on average to be considered as the least performing algorithm for iteration numbers category.

The trajectory traces of the algorithms utilised for sub-experiment 3 are shown in Figure 7-21. DA is the most consistent within its chosen path which was similar to the previous sub-experiments. CPSO and DPPSO also demonstrated consistency across ten executions in its chosen path although it is not identical to DA. CAEPSO and DACPSO are quite consistent as well with only went around the obstacle twice. The algorithm with least consistency are RRT and GA where they have the most different chosen paths within their trajectory traces. CSA is quite consistent as well but unfortunately struggling near to destination area. PRM performed quite well in this sub-experiment (as seen in Table 7-9) compared to previous experiments because of consistency showed by the algorithm in this sub-experiment where only two different paths chosen from ten runs.

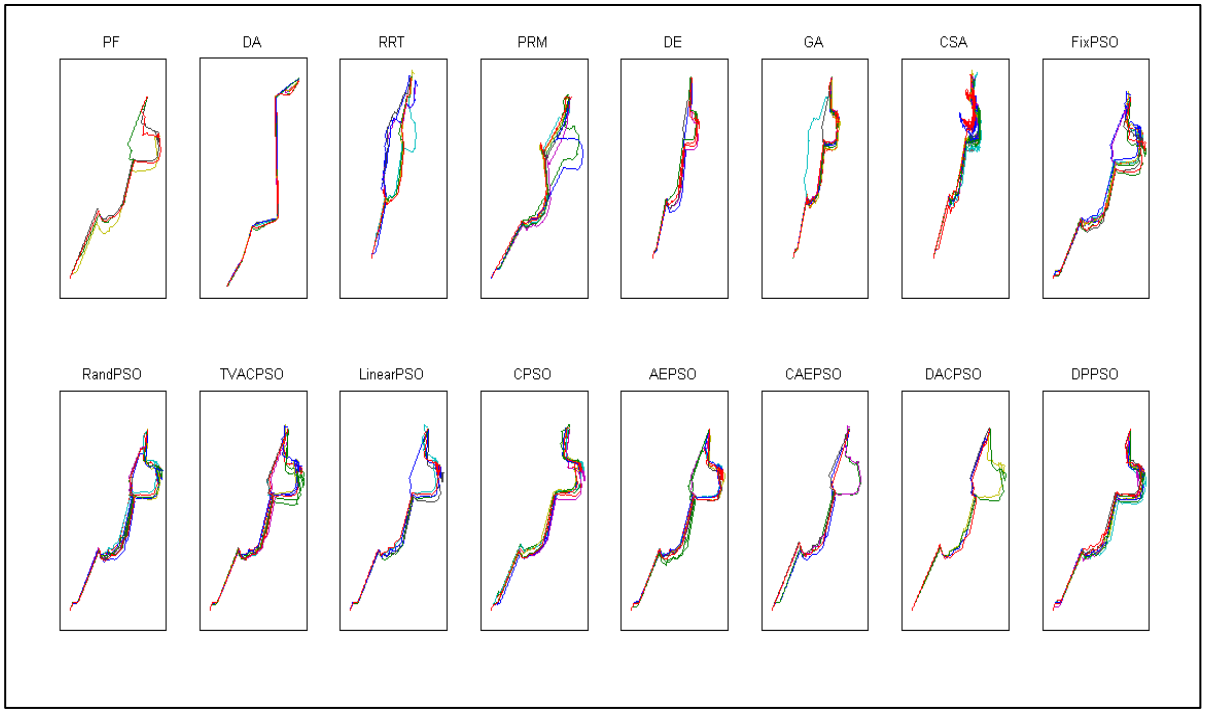


Figure 7-21. Trajectory traces of employed approaches in Path Planning Experiment for Symmetric Layout (Sub-Experiment 3). Colour variation between trajectories is indicative of different executions (trials) with maximum 10 trials.

The initial point is within region 1 while the goal point is within region 9 for this sub-experiment 3. Nine out of sixteen algorithms considered region 6 as part of their routes towards the final destination while the remaining algorithms (DA, RRT, PRM, Rand PSO, Linear PSO, CAEPSO, and DACPSO) did not consider this region within their route. With the consideration of DA as the best performing algorithm, it is notable for the algorithm that region 6 is neglected while the percentage of occupation in region 8 is quite high. DA highest occupation region is region 1 with recorded percentage of 33% where only RRT, Rand PSO, Linear PSO, CAEPSO, and DACPSO recorded the same or higher occupation percentage within this region.

The algorithm which considered region 6 as part of their route will spent a high percentage of occupation within this region. It is because this region is the trickiest area amongst

other areas because of Cul-De-Sac problem. If the algorithm can perform well, the reward is quite high, but if the algorithm cannot deal with this issue, hence it can punish with a bad result in term of execution time and total travelled distance. DPPSO could be the algorithm which benefits from performing well in this problem and become the third best performing algorithm. It is also noteworthy that the different in average travelled distance between DA-DPPSO is just 0.2860 meter. From the observation, CAEPSO and DACPSO did consider region 6 as one of their routes twice, but due to the fact that time spent here was quite short hence after rescaling, it was neglected.

CAEPSO occupied four region in total with the results of 43% for region 1, 25% for region 2, 26% of region 5 and just 5% for the destination region. DACPSO recorded 38% occupation for region 1, 24% for region 2, 28% for region 5 and 8% for region 9. Meanwhile DPPSO as third best performing algorithm considered region 1, region 2, region 5, region 6, and region 9 with the occupation percentage of 19%, 14%, 10%, 52%, and 5% respectively.

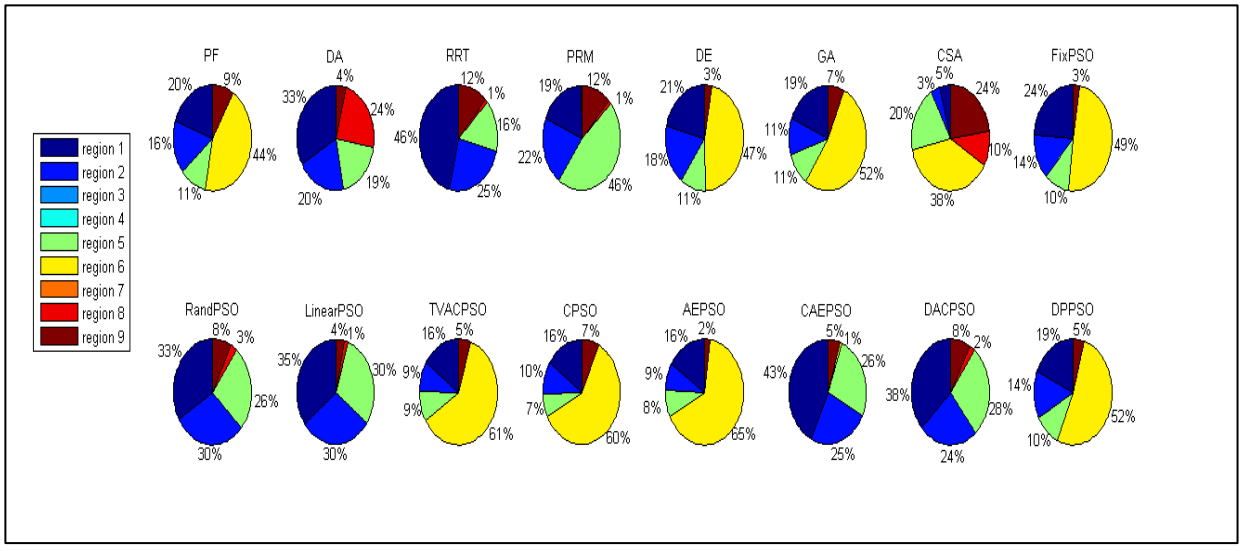


Figure 7-22. Pie Chart of Region Occupied for Dynamic Approaches PSO against other algorithms in Sub-Experiment 3 (Symmetric Layout).

Table 7-10. Overall Results for DAPSO in Path Planning Experiment on Symmetric Layout

Overall								
Algorithm	**Arrived at Destination**	**Number of Collisions**	**Displacement Problem**	**Execution Time (sec)**	**Battery Consumption (%)**	**Travelled Distance (m)**	**Convergence Iterations**	**Best Performing Algorithm**
PF	**10**	**0**	**0**	139.2497	0.5465	7.4111	-	9
DA	**10**	**0**	**0**	**85.0732**	**0.3449**	**6.8550**	-	17
RRT	**10**	**0**	**0**	364.7862	1.7359	8.2604	-	9
PRM	**10**	**0**	**0**	262.4374	0.9965	8.0963	-	9
DE	**10**	**0**	**0**	146.8259	1.4392	7.6133	473.8	9
GA	**10**	**0**	**0**	183.4562	0.8049	8.0355	510.6	9
CSA	**10**	**0**	**0**	241.1095	0.9186	8.0171	615.9	9
Fix PSO	**10**	**0**	**0**	187.1477	1.9374	7.5644	477.8	9
Rand PSO	**10**	**0**	**0**	154.4897	0.6998	7.4903	**422.9**	10
TVAC PSO	**10**	**0**	**0**	185.1215	0.8606	7.9662	461.9	10
Linear PSO	**10**	**0**	**0**	189.2685	0.8296	7.6284	459.5	9
CPSO	**10**	**0**	**0**	140.8205	0.4933	7.2104	466.8	9
AEPSO	**10**	**0**	**0**	154.6649	0.7084	7.2202	473.5	9
CAEPSO	**10**	**0**	**0**	128.3390	0.5670	6.9534	463.8	10
DACPSO	**10**	**0**	**0**	156.1648	0.5653	7.3244	479.7	9
DPPSO	**10**	**0**	**0**	126.5876	0.5151	6.9923	453.3	10

Considering all motion planning techniques performance across the three sub-experiments in this symmetric layout, DA has emerged as the best performing approach. DA is a consistent top performance on all categories considered and only outperformed once by DPPSO in the average travelled distance category for sub-experiment 1 in symmetric layout. CAEPSO and DPPSO are the second and third best performing algorithms in this experiment due to their consistently high performance in most categories across all three sub-experiments in this layout. Regarding the comparison between classical and heuristic-based algorithms, the results have shown heuristic-based methods performed quite well especially in navigation experiment for maze layout where all heuristic-based approaches managed to outperform all traditional algorithms (except DA) in total travelled distance factor. On the other hand, in navigation experiment for symmetric layout, PF managed to outperform almost all EA-based algorithms in most of the categories considered. It is notable RRT and PRM performed quite poorly especially in execution time and total travelled distance factor. Although, these methods are actually among the best path planning algorithm, they are only performing well if it is implemented for global path planning rather than local path planning.

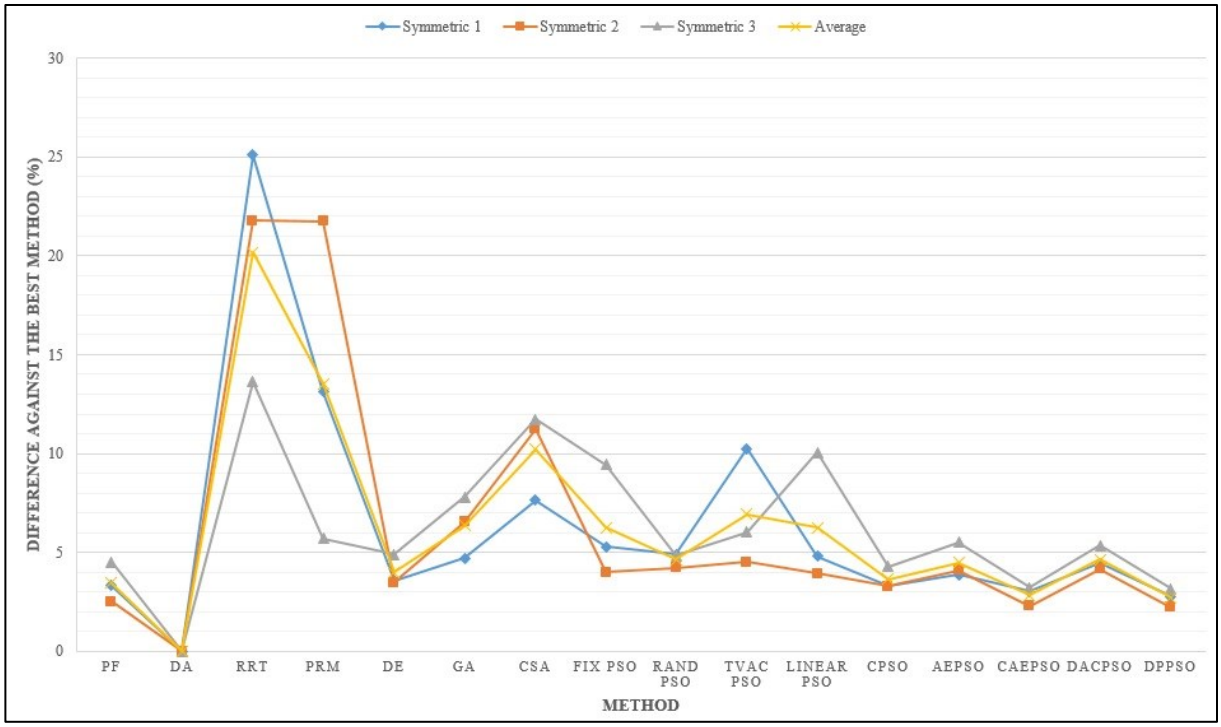


Figure 7-23. The graph of the best performing algorithm against other algorithms (Execution Time Factor).

Figure 7-23 illustrates the performance in execution time factor of each algorithm for mobile robot navigation within the symmetric layout. DA is considered as the best result that other algorithms can reach. It is because DA has the full information about the design of the environment. From the observation, CAEPSO and DPPSO are the most consistent performer with all sub-experiments recorded the difference five percent less than the result recorded by DA.

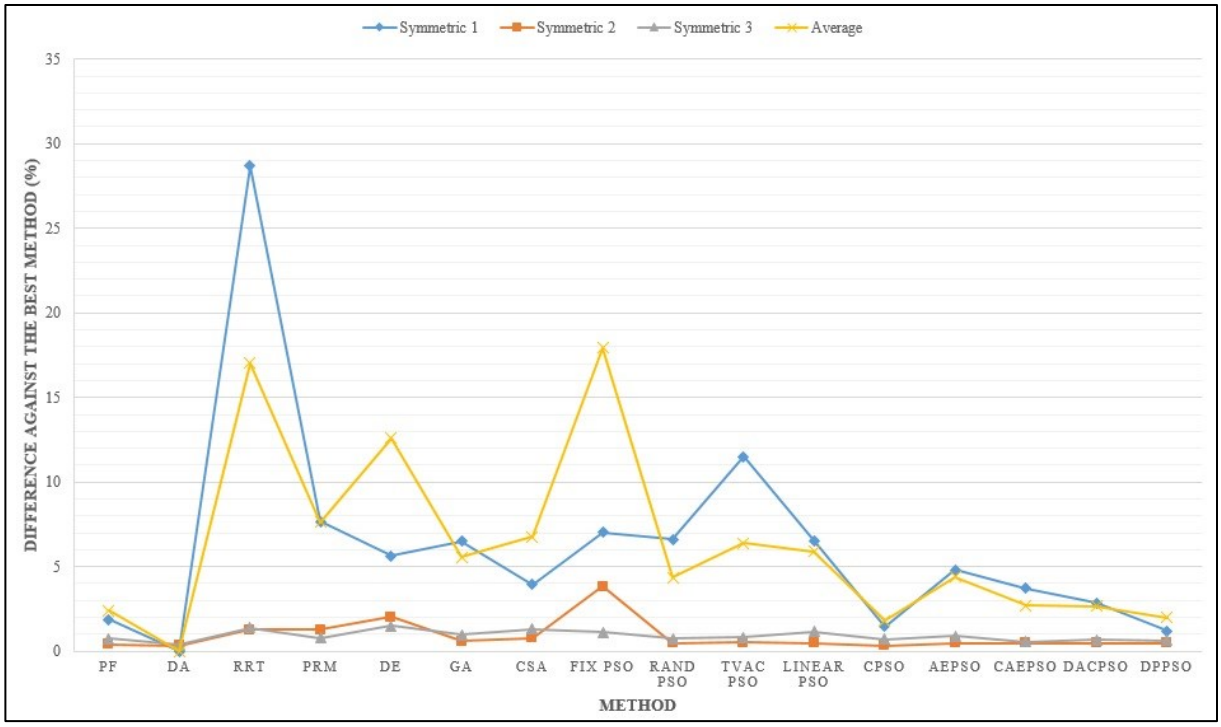


Figure 7-24. The graph of the best performing algorithm against other algorithms (Battery Consumption Factor).

Figure 7-24 shows the performance of each algorithm against the best performing algorithm for all sub-experiments involved for battery consumption factor. It is kind of interesting results as all algorithm shown the different is less than five percent for all algorithms in the second and third sub-experiments. PF, Rand PSO, CPSO, AEPSO, CAEPSO, DACPSO, and DPPSO also managed to record the different less than five percent for the first sub-experiment. The most consistent algorithms are PF, CPSO and DPPSO as all three have similar outcomes as DA.

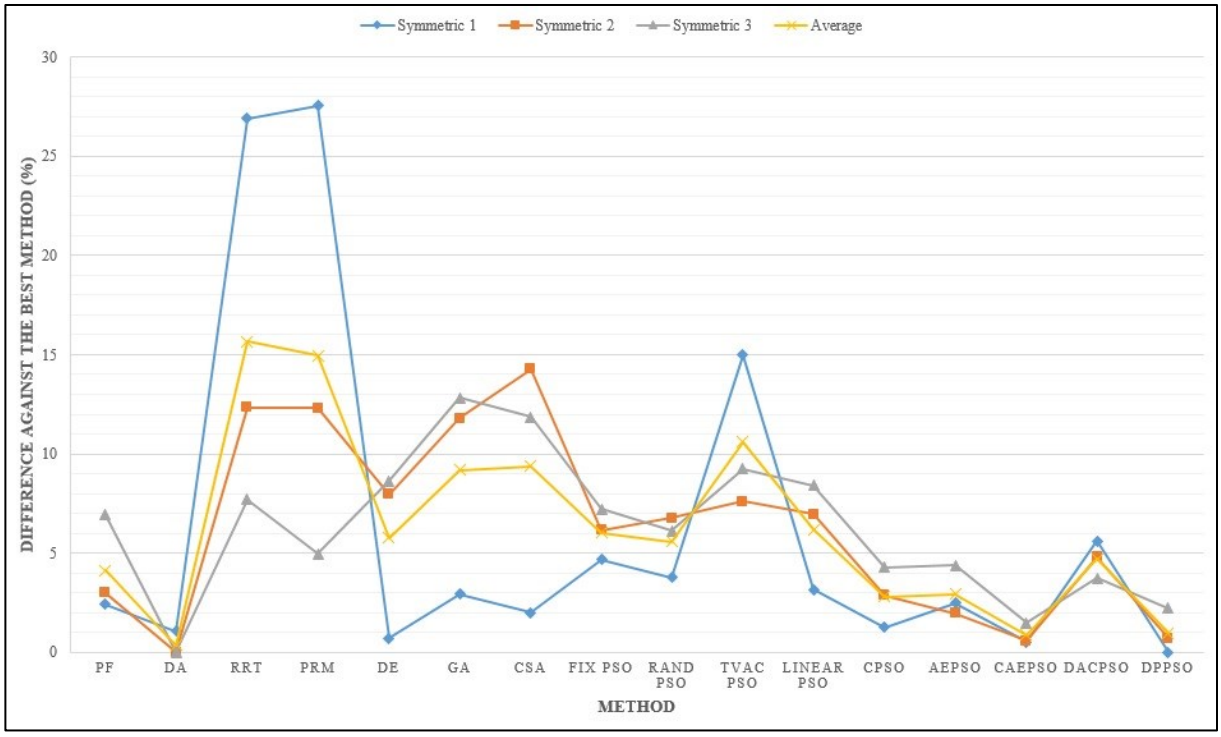


Figure 7-25. The graph of the best performing algorithm against other algorithms (Travelled Distance Factor).

Figure 7-25 illustrates the performance of all algorithms against the best performing algorithm (DA) for travelled distance category. Only four algorithms recorded results less than five percent against DA which is CPSO, AEPSO, CAEPSO, and DPPSO with the exclusion of the first sub-experiment where DPPSO is the best performing algorithm. From the observation, CAEPSO and DPPSO are recorded a result where CAEPSO shown more consistency compare to DPPSO. RRT and PRM record the worst results in the first sub-experiment with more than twenty five percent different against the best performing algorithm, DPPSO.

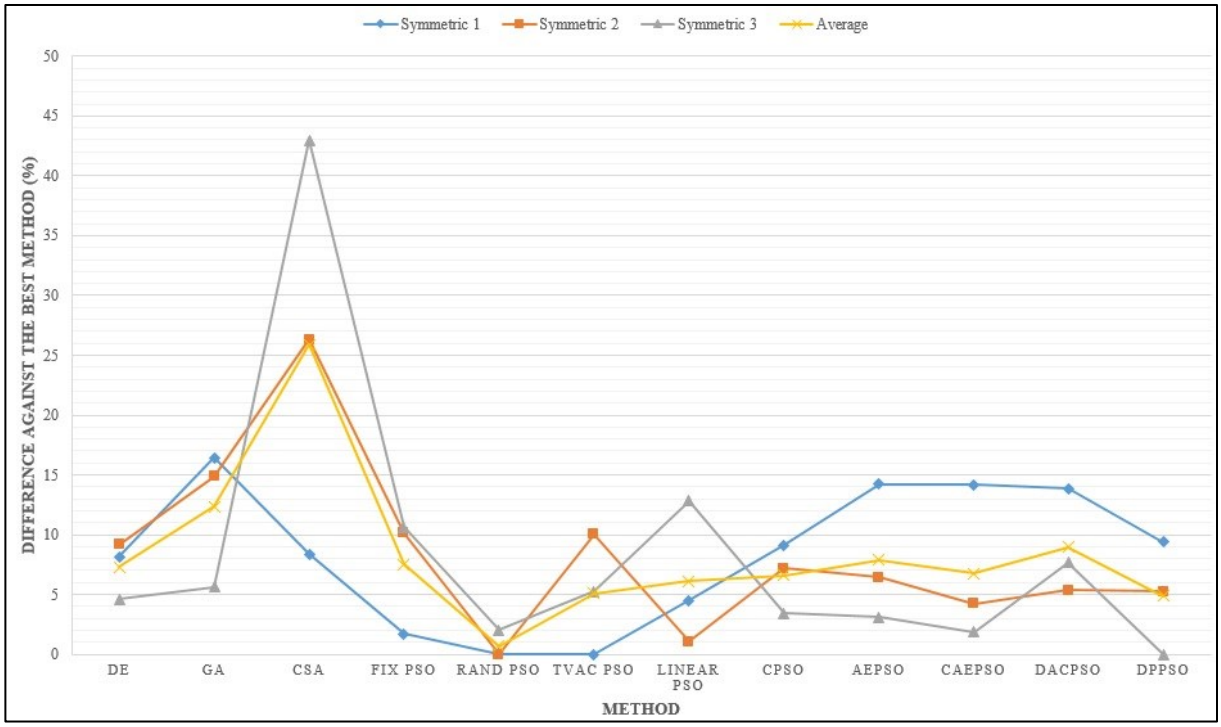


Figure 7-26. The graph of the best performing algorithm against other algorithms (Iteration Factor).

TVAC PSO, Rand PSO, and DPPSO are the best performing algorithm for sub-experiments 1, 2 and 3 respectively as shown in Figure 7-26. Rand PSO is the best performing algorithm in the first and second sub-experiments with DPPSO been the best performing algorithm in the third sub-experiments. Rand PSO is the most consistent algorithm none of its results exceed five percent different from the best performing algorithm. CSA once again shown a poor performance as two out of three results recorded are more than twenty-five percent different from the best performing method.

For comparison between classical methods, PF managed to outperform RRT and PRM in all categories. It is because PF algorithm is more compatible for local path planning compared to RRT and PRM which PRM is more direct and did not have any influence on random factor

compared to those two. Even DA managed to outperform other methods in all considered factors, it is due to the fact that DA has an advantage compared to others. Other methods do not have any knowledge about the environment layout except for the destination point which forcing them to perform local motion planning on the basis of their online sensory readings while DA had access to the complete environmental layout and performed global path planning. Thus, the results achieved by DA is reflected as the perfect optimal performance. From the overall results, DPPSO is considered as the best performing approach given that it became the second best to DA in almost every category considered in the research and closely followed by CAEPSO in the second place.

7.6 Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation

Table 7-11. Significance Analysis for Dynamic Approaches of Particle Swarm Optimisation (DAPSO).

Category	Experiments	Approaches	Experiments & Approaches
Time (s)	p = 3.35928e-40	p = 1.64714e-89	p = 5.67744e-22
Battery Consumption (%)	p = 7.22200e-01	p = 2.00000e-04	p = 3.36000e-06
Travelled Distance (m)	p = 1.13006e-68	p = 3.31946e-19	p = 1.99488e-34
Convergence iteration	p = 9.00000e-04	p = 0	p = 0

The results of statistical analysis indicated existence of significant difference between sub-experiments (p = 1.13006e-68 < 0.05), methods utilised (p = 3.3194e-19 < 0.05) and interaction between sub-experiments and the methods (p = 1.99488e-34 < 0.05). The significant of findings for the sub-experiment 1 are as follow:

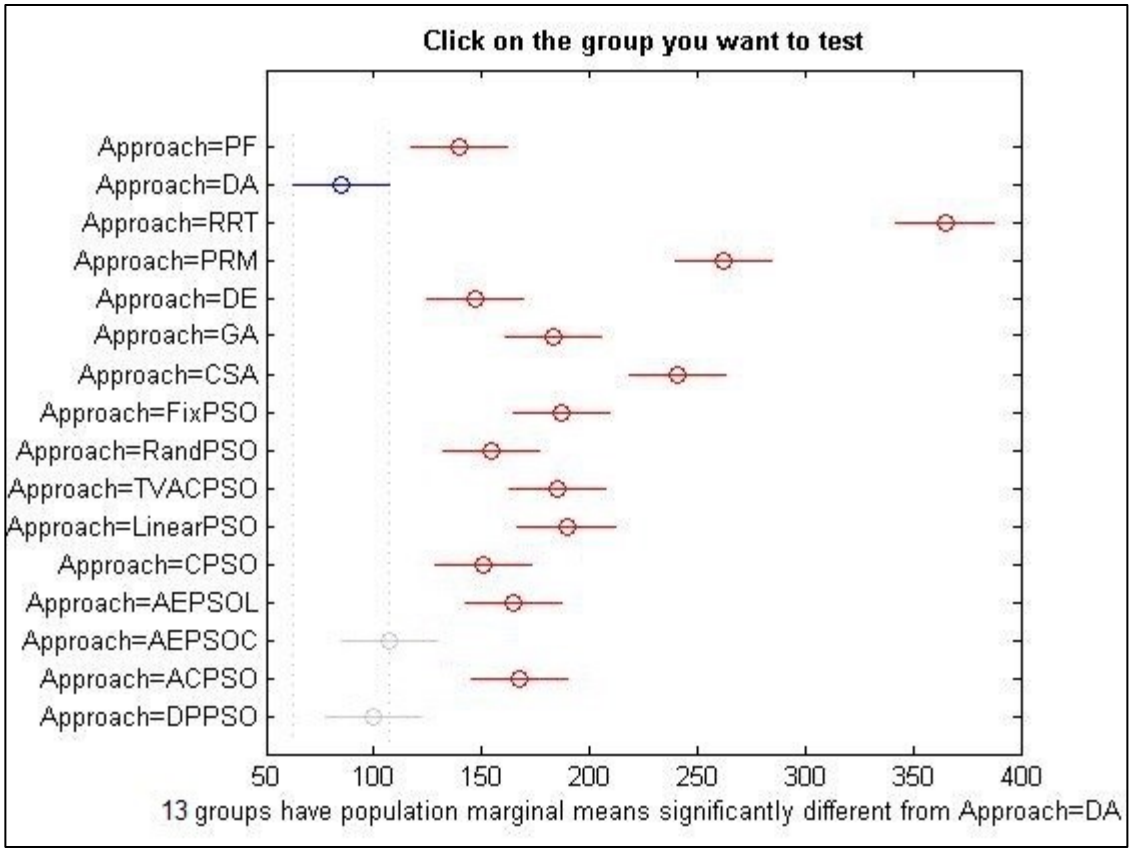


Figure 7-27. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Execution Time factor.

- Execution time (*s*): The results indicated existence of significant difference between approaches applied (p = 1.64714e-89 < 0.05). DA confirmed its superiority in this sub-experiment as best performing algorithm with significant difference against all other methods. RRT (the least performing algorithm) also shown the existence of statistical difference against all other methods. CAEPSO and DPPSO showed a lack of significant difference against PF, DE, CPSO, AEPSO and DACPSO but indicated the existence of significant difference against RRT, PRM, GA, CSA, Fix PSO, Rand PSO, TVAC PSO, and Linear PSO.

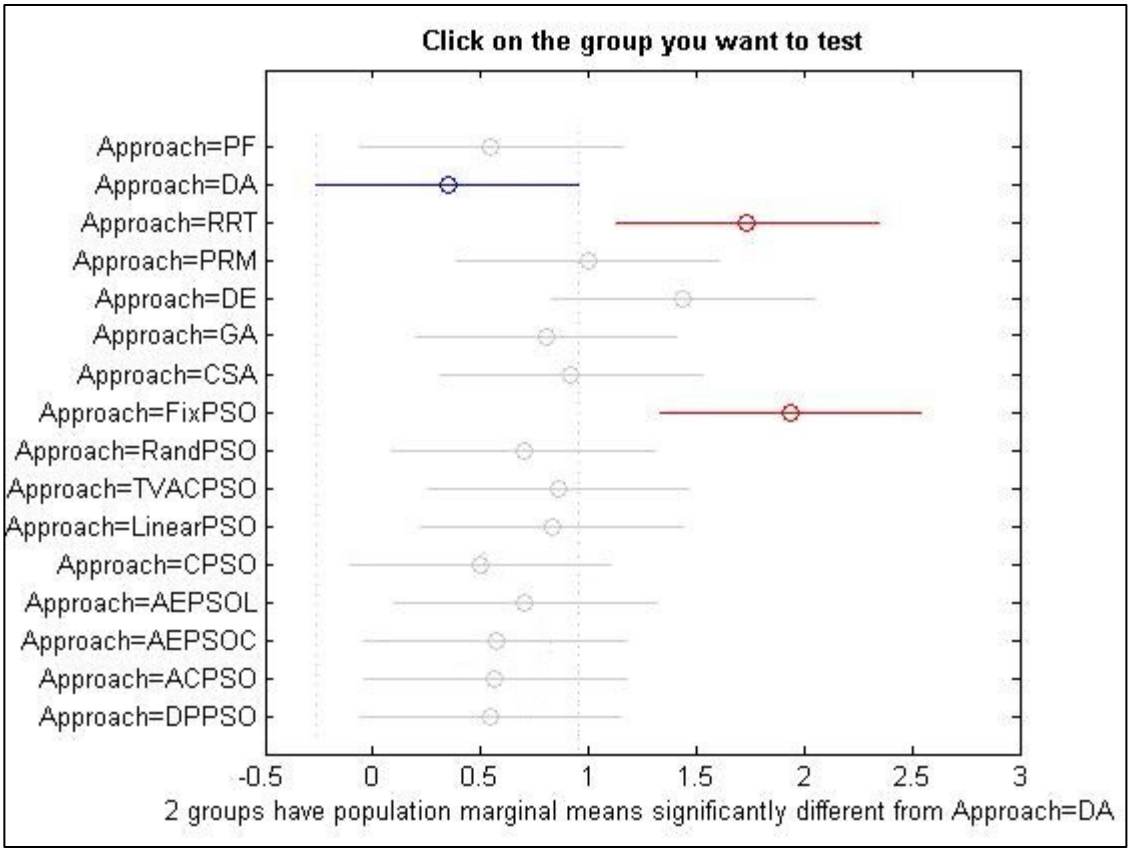


Figure 7-28. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Energy Consumption factor.

- Battery Consumption (*%*): The results shown there is significant difference between algorithms for this category (p = 0.002 < 0.05). Figure 7-28 illustrates DA shown existence of significant difference against RRT, and Fix PSO. PF, CPSO, AEPSO, ACPSO, and DPPSO showed significant difference against Fix PSO. Meanwhile, the other algorithms indicated a lack of significant difference amongst themselves.

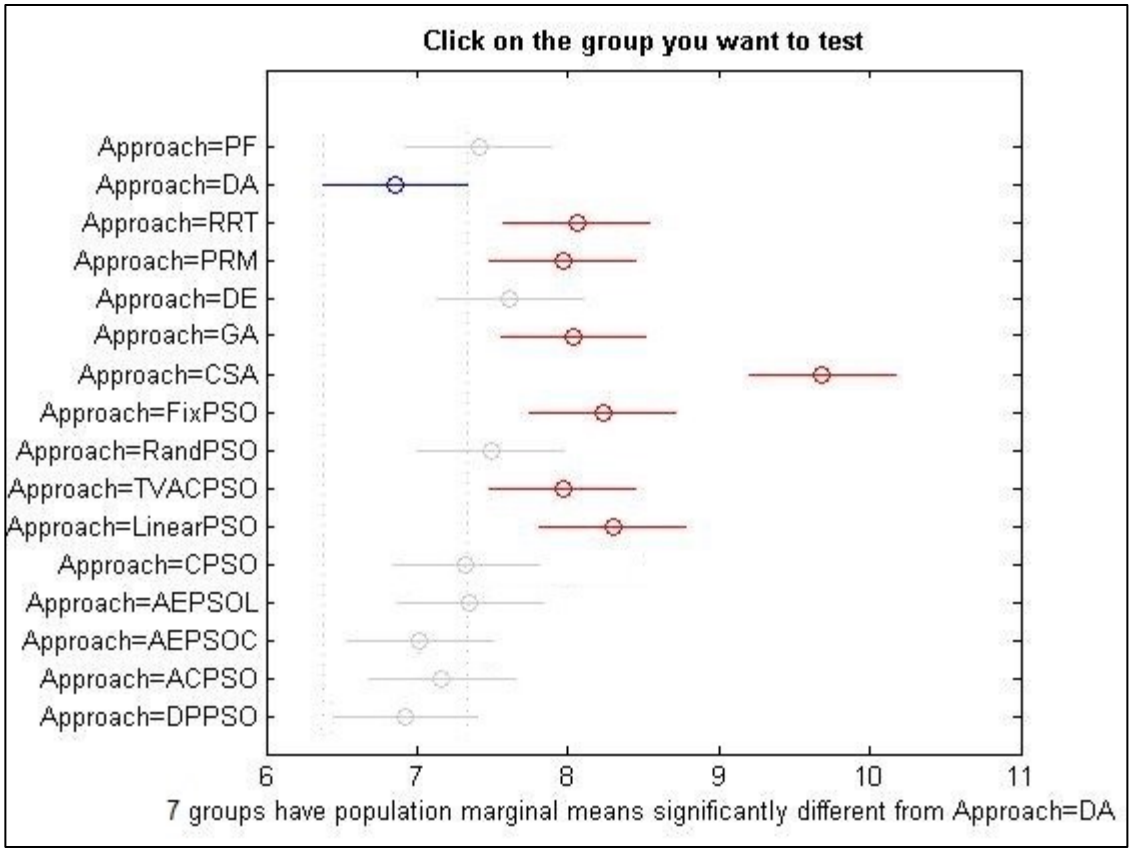


Figure 7-29. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Travelled Distance factor.

- Total travelled distance (*m*): The findings shown existence of significant difference amongst algorithms (p = 3.31946e-19 < 0.05). Figure 7-29 illustrates DA, CAEPSO, and DPPSO showed significant difference against RRT, PRM, GA, CSA, Fix PSO, TVAC PSO, and Linear PSO. PF, DE, Rand PSO, CPSO, DACPSO, and AEPSO only shown significant difference against CSA but indicated a lack of statistical difference against other algorithms.

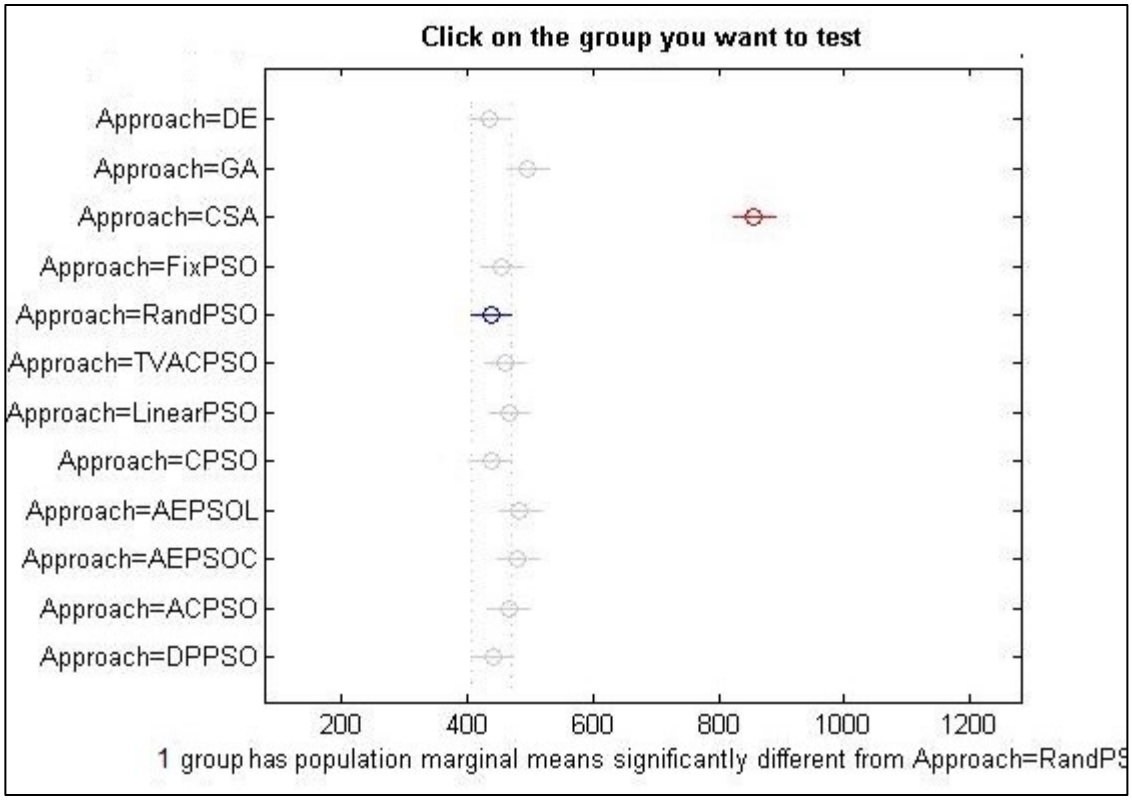


Figure 7-30. Box Plot of Significant Difference in Mobile Robot Navigation Problem (Symmetric Layout) for Iteration factor.

- Convergence iteration: The results of significant analysis shown existence of significant difference for this considered factor ($p = 0.00 < 0.05$). Figure 7-30 shows all algorithms shown significant difference CSA. Meanwhile, the other algorithms are shown insignificant difference among themselves.

7.7 Summary

This chapter has discussed the results achieved from two set of layouts for mobile robot navigation experiment. Two types of proposed PSO (Morphology based PSO and Dynamic Behaviours based PSO) were tested against twelve algorithms and thirteen algorithms respectively (including four classical path planning methods). From the observation, discussion,

and analysis, all newly introduced PSOs are considered as encouraging especially for MPSO, CAEPSO, and DPPSO where they managed to outperform other evolutionary algorithms (except DA, due to its advantages) almost easily. These three algorithms are outstanding especially sub-experiment 1 for symmetric layout where they managed to outperform DA for total travelled distance category although they did not have any pieces of knowledge about the map design. From the findings as well, these algorithms shown a lack of significant different against the best performing algorithm for all categories considered which shown their competitiveness even though they did not have any information about the layouts.

CHAPTER 8

CONCLUSIONS AND FUTURE WORKS

8.1 Conclusion

Particle Swarm Optimisation (PSO) belongs as one of the evolutionary based algorithms, which is a population-based stochastic algorithm and a popular option in the past years for solving any types of optimisation problems as it has excellent capabilities to overcome those problems. However, despite the popularity of this approach, with most of the optimisation problems that have different dimension and requirement for parameter settings, it has two major drawbacks. The first drawback is the suitable limitation or boundaries applied for particles in the algorithm. The other is the parameter settings.

This research attempts to solve these two major problems which are the compatibility of PSO to be implemented onto the online application and the adaptable evolutionary techniques to any optimisation problems. Hence, the first two techniques are focusing on tackling the first problem mentioned above while the other three techniques concentrating on another problem. Morphology PSO and Constricted MPSO do not only manage to outperform (or at least par) the other algorithms but also manage to give decent outcome as well in both simulation and real-world environment as seen in all experiments involved. The main goal is to find a technique which is not only adaptable to any optimisation problems but can converge towards the global optimal within complex circumstances as well. Not only on simulation

platform but in the real-world platform too. Hence, these three dynamic behaviour PSOs named as Constricted Area Extended PSO (CAEPSO), Dynamic Acceleration Coefficients PSO (DACPSO) and Dynamic Parameterising PSO (DPPSO) have been proposed. This dynamic behaviour is not only giving those three algorithms adaptability skills but also the ability to perform splendidly in online application as well as evidence shown in mobile robot navigation problem.

Three main experiments had been designed to assess the performance of five proposed methods against existing evolutionary algorithms including four classical methods for mobile robot navigation experiment. In the first experiment (the benchmark function), focusing on Morphology based PSO, both proposed algorithms performed quite well, especially for MPSO. CMPSO might not perform well as MPSO due to the constriction factor that was added to velocity equation. This constriction effect reduced almost 30% of the actual value obtained from velocity equation which was the step size for the particle. Hence, this probably affects the exploration behaviour of the swarm at the beginning of iterations and as a result, the global optimal outcome cannot be achieved in the end. The same result was achieved in the second experiment (engineering design) where MPSO became the best performing algorithm compared to the other EA-based methods. MPSO also managed to become overall the best performing algorithm compared to the other EA-based methods including several classical path planning methods (with an exceptional of Dijkstra's Algorithm).

The three dynamic based PSO were outstanding regarding performance especially CAEPSO and DPPSO throughout all experiments involved. In benchmark function experiment, DPPSO was the best overall performing algorithm with CAEPSO in second with only one benchmark function short. Both of them shared the top spot as the best performing algorithm

for engineering design optimisation problems. Meanwhile, DACPSO's overall outcomes were not far off from CAEPSO and DPPSO in this type of optimisation problems. CAEPSO and DPPSO once again managed to dominate all categories considered for mobile robot navigation problems with an exemption of Dijkstra's Algorithm since DA has an advantage compare to the other algorithms. Hence, it is seen as the best results can be achieved by the other algorithms.

This research has also verified several studies done previously. This research verified a research which stated the Differential Evolution is performing better than Genetic Algorithm (Ab Wahab et al., 2015; Dong, Liu, Tao, Li, & Xin, 2012). On the one hand, this research confirmed the superiority of Constricted PSO against Linear PSO (Bai, 2010; Parsopoulos & Vrahatis, 2011; Syed Abdullah et al., 2012; Trelea, 2003; H. Zhu et al., 2013) concerning overall performances. On the other hand, it has also certified Linear PSO is superior compared to the basic PSO (Fix PSO) regarding performance (Arasomwan & Adewumi, 2013).

In conclusion, all objectives for this research have been achieved, and it also managed to solve the two main problems commonly faced by researchers in swarm intelligence field. However, these approaches are not limited to PSO algorithm only as it is not specifically designed to only one evolutionary algorithm. Therefore, these approaches can be implemented to any evolutionary algorithms as long as the right elements are manipulated to achieve these types of approaches.

8.2 Future Work

This research has introduced two new methods for a PSO-based algorithm for solving both simulation and online optimisation problems. It will be interesting to investigate further to see

the capabilities and performances of these five new proposed algorithms. Here are several suggestions that can be looked into to assess their limits:

1. **Implement the proposed methods on the online application using microscopic approach.**

   In the third experiment, the online application is used as a platform to the capabilities of each algorithms utilised. For EA-based algorithms, the macroscopic approach is employed instead of microscopic because of several constraints. Therefore, it will be appealing to see the behaviour and results of each EA-based methods when it is implemented using microscopic approach for mobile robot navigation problems.

2. **Fine tuning the a_c and a_p value for better results.**

   The current values used for a_c and a_p are 0.005. This value is obtained from several different values tested on multiple benchmark functions which can be referred in Appendix B. However, more precise value of a_c and a_p can be acquired using any optimisation algorithm such as DE, CSA or PSO.

3. **Use different approaches for Dynamic Acceleration Coefficients PSO.**

   Dynamic Acceleration Coefficients PSO (DACPSO) has shown quite decent overall performances compared to Linear PSO. However, it is not as good as MPSO, CAEPSO or DPPSO. Perhaps the fix awarding population approach is not quite efficient to enhance the overall performance of PSO. Therefore, different approaches such as high awarding population or low awarding population approach can be used to give a better performance in the end for DACPSO.

4. **Test on higher and lower iteration.**

This research utilised a thousand iterations for each EA-based algorithms involved including the newly proposed PSO. It will be interesting to observe the effect of lower and higher iteration towards these five newly proposed PSO regarding behaviour and final outcomes.

5. **Test on bigger and smaller population size.**

   One of the variables that can affect the outcome of the optimisation problem is the population size. Small population size can lead to premature convergence, and massive population size can result in an excessive computing power. This research utilised population size of hundred which aimed to make it more challenging for the EA-based algorithm to find the best result with a limited population. Hence, it will be fascinating to investigate the consequence of smaller and bigger population size towards all five proposed methods.

6. **Combining MPSO with AEPSO.**

   MPSO and AEPSO were two variants of PSO in this research which showed promising results throughout all experiments involved. However, these two variants have a different type of approaches where AEPSO is using sub-swarm while MPSO is only using single level swarm. Therefore, the combination of MPSO and AEPSO are predicted to give a better outcome for the optimisation problem.

REFERENCES

Ab Wahab, M. N., Nefti-Meziani, S., & Atyabi, A. (2015). A comprehensive review of swarm optimisation algorithms. *PLoS ONE*, *10*(5). http://doi.org/10.1371/journal.pone.0122827

Abdallah, H., Emara, H. M., Dorrah, H. T., & Bahgat, A. (2009). Using Ant Colony Optimization algorithm for solving project management problems. *Expert Systems with Applications*, *36*(6), 10004–10015. http://doi.org/10.1016/j.eswa.2008.12.064

Abu-Dakka, F. J., Valero, F., & Mata, V. (2012). Evolutionary Path Planning Algorithm for Industrial Robots. *Advanced Robotics*, *26*(11–12), 1369–1392. http://doi.org/10.1080/01691864.2012.689743

Alam, M. S., & Rafique, M. U. (2015). Mobile robot path planning in environments cluttered with non-convex obstacles using particle swarm optimization. *Control, Automation and Robotics (ICCAR), 2015 International Conference on*. http://doi.org/10.1109/ICCAR.2015.7165997

AlRashidi, M. R., & El-Hawary, M. E. (2009). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, *13*(4), 913–918. http://doi.org/10.1109/TEVC.2006.880326

Araghi, S., Khosravi, A., & Creighton, D. (2015). Intelligent cuckoo search optimized traffic signal controllers for multi-intersection network. *Expert Systems with Applications*, *42*(9), 4422–4431. http://doi.org/10.1016/j.eswa.2015.01.063

Atyabi, A., & Phon-Amnuaisuk, S. (2007). Particle swarm optimization with area extension (AEPSO). In *2007 IEEE Congress on Evolutionary Computation, CEC 2007* (pp. 1970–

1976). http://doi.org/10.1109/CEC.2007.4424715

Atyabi, A., Phon-Amnuaisuk, S., & Ho, C. K. (2010). Applying area extension PSO in robotic swarm. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *58*, 253–285. http://doi.org/10.1007/s10846-009-9374-2

Atyabi, A., & Powers, D. M. W. (2010). The use of Area Extended Particle Swarm Optimization (AEPSO) in swarm robotics. In *11th International Conference on Control, Automation, Robotics and Vision, ICARCV 2010* (pp. 591–596). http://doi.org/10.1109/ICARCV.2010.5707854

Atyabi, A., & Powers, D. M. W. (2013). Cooperative area extension of PSO: Transfer learning vs. uncertainty in a simulated swarm robotics. In *ICINCO 2013 - Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (Vol. 1, pp. 177–184).

Atyabi, A., & Powers, D. M. W. (2013). Review of classical and heuristic-based navigation and path planning approaches. *International Journal of Advancements in Computing Technology (IJACT)*, *5*(14), 1–14.

Bai, Q. (2010). Analysis of Particle Swarm Optimization Algorithm. *Computer and Information Science*, *3*(1), 180–184. http://doi.org/10.5539/cis.v3n1P180

Balaprakash, P., & Birattari, M. (2006). Incremental local search in ant colony optimization: Why it fails for the quadratic assignment problem. *Ant Colony Optimization and Swarm Intelligence*, 156–166.

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. Part

I: Background and development. *Natural Computing*. http://doi.org/10.1007/s11047-007-9049-5

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. In *Natural Computing* (Vol. 7, pp. 109–124). http://doi.org/10.1007/s11047-007-9050-z

Barraquand, J., Langlois, B., & Latombe, J.-C. (1992). Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, *22*(2), 224–241. http://doi.org/10.1109/21.148426

Basu, M., & Chowdhury, A. (2013). Cuckoo search algorithm for economic dispatch. *IET Generation, Transmission & Distribution*, *60*(February), 99–108. http://doi.org/10.1016/j.energy.2013.07.011

Baterina, A. V., & Oppus, C. (2010). Image edge detection using ant colony optimization. *WSEAS Transactions on Signal Processing*, *6*(2), 58–67. http://doi.org/10.4156/jdcta.vol6.issue11.24

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, *2*(4), 353–373. http://doi.org/10.1016/j.plrev.2005.10.001

Blum, C., & Li, X. (2008). Swarm Intelligence in Optimization. *Swarm Intelligence Introduction and Applications*, 43–85. http://doi.org/10.1007/978-3-540-74089-6

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: from natural to artificial systems*. *Handbook of Nature-Inspired and Innovative Computing*.

http://doi.org/10.1007/s13398-014-0173-7.2

Bratton, D., & Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium, SIS 2007* (pp. 120–127). http://doi.org/10.1109/SIS.2007.368035

Brezina Jr Zuzana Čičková, I. (2011). Solving the Travelling Salesman Problem Using the Ant Colony Optimization. *Management Information Systems*, *6*(4), 10–14.

Busch, J., Quirico, M., Richter, L., Schmidt, M., Raatz, A., & Nyhuis, P. (2015). A genetic algorithm for a self-learning parameterization of an aerodynamic part feeding system for high-speed assembly. *CIRP Annals - Manufacturing Technology*, *64*(1), 5–8. http://doi.org/10.1016/j.cirp.2015.04.044

Cagnina, L. C., Esquivel, S. C., & Coello Coello, C. A. (2008). Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica (Ljubljana)*, *32*(3), 319–326.

Cen, Y. C. Y., Song, C. S. C., Xie, N. X. N., & Wang, L. W. L. (2008). Path planning method for mobile robot based on ant colony optimization algorithm. *2008 3rd IEEE Conference on Industrial Electronics and Applications*, 298–301. http://doi.org/10.1109/ICIEA.2008.4582528

Cesmeci, D., & Gullu, M. K. (2010). Sub-swarm converging linear particle swarm optimization. In *Signal Processing and Communications Applications Conference (SIU), 2010 IEEE 18th* (pp. 736–739).

Chang, Y., & Yu, G. (2013). Multi-Sub-Swarm PSO Classifier Design and Rule Extraction.

International Workshop on Cloud Computing and Information Security, 104–107.

Chaowanawatee, K., & Heednacram, A. (2012). Implementation of Cuckoo Search in RBF Neural Network for Flood Forecasting. *2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks*, 22–26. http://doi.org/10.1109/CICSyN.2012.15

Chen, Y. B., Yu, J. Q., Su, X. L., & Luo, G. C. (2014). Path Planning for Multi-UAV Formation. *Journal of Intelligent and Robotic Systems: Theory and Applications*, *77*(1), 229–246. http://doi.org/10.1007/s10846-014-0077-y

Cheng, G., & Zelinsky, A. (1995). A Physically Grounded Search in a Behaviour Based Robot. In *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence* (pp. 547–554).

Chernbumroong, S., Cang, S., & Yu, H. (2015). Genetic algorithm-based classifiers fusion for multisensor activity recognition of elderly people. *IEEE Journal of Biomedical and Health Informatics*, *19*(1), 282–289. http://doi.org/10.1109/JBHI.2014.2313473

Chia, S.-H., Su, K.-L., Guo, J.-H., & Chung, C.-Y. (2010). Ant Colony System Based Mobile Robot Path Planning. *2010 Fourth International Conference on Genetic and Evolutionary Computing*, 210–213. http://doi.org/10.1109/ICGEC.2010.59

Chiao Mei, F. C., Phon-Amnuaisuk, S., Alias, M. Y., & Leong, P. W. (2008). Adaptive GA: An essential ingredient in high-level synthesis. In *2008 IEEE Congress on Evolutionary Computation, CEC 2008* (pp. 3837–3844). http://doi.org/10.1109/CEC.2008.4631319

Chiu, C., Cheng, Y., & Chang, C. (2012). Comparison of Particle Swarm Optimization and

Genetic Algorithm for the Path Loss Reduction in an Urban Area. *Journal of Applied Science and ...*, *15*(4), 371–380.

Choi, Y., & Chang, D. (2013). Development of pressure vessel design process using optimization and structural reliability method. In *Safety, Reliability, Risk and Life-Cycle Performance of Structures and Infrastructures - Proceedings of the 11th International Conference on Structural Safety and Reliability, ICOSSAR 2013* (pp. 4931–4934).

Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73. http://doi.org/10.1109/4235.985692

Contreras-Cruz, M. a., Ayala-Ramirez, V., & Hernandez-Belmonte, U. H. (2015). Mobile robot path planning using artificial bee colony and evolutionary programming. *Applied Soft Computing*, *30*, 319–328. http://doi.org/10.1016/j.asoc.2015.01.067

Das, S., Nagaratnam Suganthan, P., & Member, S. (2011). Differential Evolution_A Practical Approach to Global Optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, *15*(1). http://doi.org/10.1109/TEVC.2010.2059031

De Jong, K. A., & Spears, W. M. (1992). A formal analysis of the role of multi-point crossover in genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, *5*(1), 1–26. http://doi.org/10.1007/BF01530777

De Medeiros, L. F. (2015). Using genetic algorithm for calculating the production mix in business plan simulator . *Gestao E Producao*, *22*(3), 624–635. http://doi.org/10.1590/0104-530X142-12

Deep, K., & Singh, P. K. (2015). Design of robust cellular manufacturing system for dynamic part population considering multiple processing routes using genetic algorithm. *Journal of Manufacturing Systems*, *35*, 155–163. http://doi.org/10.1016/j.jmsy.2014.09.008

Demirel, N. Ç., & Toksarı, M. D. (2006). Optimization of the quadratic assignment problem using an ant colony algorithm. *Applied Mathematics and Computation*, *183*(1), 427–435. http://doi.org/10.1016/j.amc.2006.05.073

Dijkstra, E. W. (1959). A note on two problems in connection with graphs. *Numerische Mathematik*, *1*, 269–271. http://doi.org/10.1007/BF01386390

Donglan, Z. (2014). Research on the Sensor Coarse Signal Processing Model Based on Adaptive Genetic Algorithm. In *COMPUTER-AIDED DESIGN, MANUFACTURING, MODELING AND SIMULATION III* (Vol. 443, pp. 342–345). http://doi.org/10.4028/www.scientific.net/AMM.443.342

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. http://doi.org/10.1109/MCI.2006.329691

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*(2–3), 243–278. http://doi.org/10.1016/j.tcs.2005.05.020

Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 1–24.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. *Mendeley Desktop*. http://doi.org/10.4249/scholarpedia.1461

Eberhart, R. C., & Shi, Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization.pdf. In *the Congress on Evolutionary Computation 2000* (pp. 84–88). http://doi.org/10.1109/CEC.2000.870279

Elbanhawi, M., & Simic, M. (2014). Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, *2*, 56–77. http://doi.org/10.1109/ACCESS.2014.2302442

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics*, *19*(1), 43–53. http://doi.org/10.1016/j.aei.2005.01.004

Engelbrecht, A. P. (2010). Heterogeneous Particle Swarm Optimization. In *Swarm Intelligence* (Vol. 6234, pp. 191–202). http://doi.org/10.1007/978-3-642-15461-4_17

Engelbrecht, A. P. (2011). Scalability of a heterogeneous particle swarm optimizer. In *IEEE SSCI 2011 - Symposium Series on Computational Intelligence - SIS 2011: 2011 IEEE Symposium on Swarm Intelligence* (pp. 1–8). http://doi.org/10.1109/SIS.2011.5952563

Figueiredo, E. M. N., & Ludermir, T. B. (2012). Effect of the PSO topologies on the performance of the PSO-ELM. In *Proceedings - Brazilian Symposium on Neural Networks, SBRN* (pp. 178–183). http://doi.org/10.1109/SBRN.2012.26

Fleetwood, K. (1999). An Introduction to Differential Evolution. *New Ideas in Optimization*, 79–108. http://doi.org/10.1038/155531c0

Fong, S., Wong, R., & Vasilakos, A. V. (2016). Accelerated PSO Swarm Search Feature Selection for Data Stream Mining Big Data. *IEEE Transactions on Services Computing*, *9*(1), 33–45. http://doi.org/10.1109/TSC.2015.2439695

Gao, H., Kwong, S., Yang, J., & Cao, J. (2013). Particle swarm optimization based on intermediate disturbance strategy algorithm and its application in multi-threshold image segmentation. *Information Sciences*, *250*, 82–112. http://doi.org/10.1016/j.ins.2013.07.005

Geraerts, R., & Overmars, M. H. (2004). A comparative study of probabilistic roadmap planners. In *Springer Tracts in Advanced Robotics* (Vol. 7 STAR, pp. 43–57). http://doi.org/10.1007/978-3-540-45058-0_4

Gupta, S., Kumar, V., & Agarwal, G. (2010). Task Scheduling in Multiprocessor System Using Genetic Algorithm. In *2010 Second International Conference on Machine Learning and Computing* (pp. 267–271). http://doi.org/10.1109/ICMLC.2010.50

Han, W., Zhang, X., Jiang, H., & Li, W. (2014). An Ant Colony Optimization Algorithm for Software Project Management. *Control and Automation (CA), 2014 7th Conference on*. http://doi.org/10.1109/CA.2014.12

He, Y., Zeng, Q., Liu, J., Xu, G., & Deng, X. (2013). Path planning for indoor UAV based on Ant Colony Optimization. *2013 25th Chinese Control and Decision Conference (CCDC)*, 2919–2923. http://doi.org/10.1109/CCDC.2013.6561444

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. *Ann Arbor MI University of Michigan Press* (Vol. Ann Arbor). http://doi.org/10.1137/1018105

Hwang, Y. K., & Ahuja, N. (1992). A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, *8*(1), 23–32. http://doi.org/10.1109/70.127236

Imran, M., Hashim, R., & Khalid, N. E. A. (2013). An overview of particle swarm optimization

variants. In *Procedia Engineering* (Vol. 53, pp. 491–496). http://doi.org/10.1016/j.proeng.2013.02.063

Iqbal, N., Zerguine, A., & Al-Dhahir, N. (2015). Decision Feedback Equalization using Particle Swarm Optimization. *Signal Processing*, *108*, 1–12. http://doi.org/10.1016/j.sigpro.2014.07.030

Jahanzaib, M., Masood, S. A., Nadeem, S., Akhtar, K., & Shahbaz, M. (2012). Application of Genetic Algorithm (GA) approach in the formation of manufacturing cells for group technology. *Life Science Journal*, *9*(4), 799–809.

Jaillet, L., & Porta, J. M. (2013). Path planning under kinematic constraints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, *29*(1), 105–117. http://doi.org/10.1109/TRO.2012.2222272

Junjie, P. J. P., & Dingwei, W. D. W. (2006). An Ant Colony Optimization Algorithm for Multiple Travelling Salesman Problem. *First International Conference on Innovative Computing, Information and Control - Volume I (ICICIC'06)*, *1*. http://doi.org/10.1109/ICICIC.2006.40

Kavraki, L. E., Kavraki, L. E., Svestka, P., Svestka, P., Latombe, J.-C., Latombe, J.-C., … Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *Robotics and Automation, IEEE Transactions on*, *12*(4), 566–580. http://doi.org/10.1109/70.508439

Kennedy, J. (2006). Swarm intelligence. *Handbook of Nature-Inspired and Innovative Computing*, 187–219. http://doi.org/10.1007/0-387-27705-6_6

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (Vol. 4, pp. 1942–1948 vol.4). http://doi.org/10.1109/ICNN.1995.488968

Khatib, O. (1986). Real time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics and Research*. http://doi.org/10.1177/027836498600500106

Kia, R., Khaksar-Haghani, F., Javadian, N., & Tavakkoli-Moghaddam, R. (2014). Solving a multi-floor layout design model of a dynamic cellular manufacturing system by an efficient genetic algorithm. *Journal of Manufacturing Systems*, *33*(1), 218–232. http://doi.org/10.1016/j.jmsy.2013.12.005

Kim, D. K., Jeong, K. S., McKay, R. I. B., Chon, T. S., & Joo, G. J. (2012). Machine learning for predictive management: Short and long term prediction of phytoplankton biomass using genetic algorithm based recurrent neural networks. *International Journal of Environmental Research*, *6*(1), 95–108.

Kiranyaz, S., Ince, T., Yildirim, A., & Gabbouj, M. (2010). Fractional particle swarm optimization in multidimensional search space. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *40*(2), 298–319. http://doi.org/10.1109/TSMCB.2009.2015054

Kuffner, J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. *Proc. IEEE International Conference on Robotics and Automation ICRA '00*, *2*(Icra), 995--1001 vol.2. http://doi.org/10.1109/ROBOT.2000.844730

Kumar, A., & Chakarverty, S. (2011). Design optimization for reliable embedded system using

Cuckoo search. In *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology* (Vol. 1, pp. 264–268). http://doi.org/10.1109/ICECTECH.2011.5941602

Kumar, M., Husian, M., Upreti, N., & Gupta, D. (2010). Genetic Algorithm: Review and Application. *International Journal of Information Technology and Knowledge Management*, *2*(2), 451–454.

Lan, T., & Li, W. (2010). A macroscopic state model of swarm robot system. In *Proceedings - 2010 2nd WRI Global Congress on Intelligent Systems, GCIS 2010* (Vol. 2, pp. 258–261). http://doi.org/10.1109/GCIS.2010.239

Langeveld, J. (2011). A Generic Set-Based Particle Swarm Optimization Algorithm. In *ICSI 2011: International conference on swarm intelligence* (pp. 1–10).

LaValle, S. M. (1998). Rapidly-Exploring Random Trees: A New Tool for Path Planning. *In*, *129*, 98–11. http://doi.org/10.1.1.35.1853

LaValle, S. M. (2011). Motion Planning. Part II: Wild Frontiers. *IEEE Robotics & Automation Magazine*, *18*(June), 108–118. http://doi.org/10.1109/MRA.2011.941635

Layeb, A. (2011). A novel quantum inspired cuckoo search for knapsack problems. *International Journal of Bio-Inspired Computation*, *3*(5), 297. http://doi.org/10.1504/IJBIC.2011.042260

Lerman, K., Martinoli, A., & Galstyan, A. (2005). A Review of Probabilistic Macroscopic Models for Swarm Robotic Systems. In *Swarm Robotics* (Vol. 3342, pp. 143–152). http://doi.org/10.1007/978-3-540-30552-1_12

Li, T., Chen, W., Zheng, X., & Zhang, Z. (2009). An improvement of the ant colony optimization algorithm for solving travelling salesman problem (TSP). *Wireless Communications, Networking and Mobile ...*, 1–3. http://doi.org/10.1109/WICOM.2009.5302937

Li, Y., & Chen, Y. (2010). A genetic algorithm for job-shop scheduling. *Journal of Software*, *5*(3), 269–274. http://doi.org/10.4304/jsw.5.3.269-274

Lihong, Q., & Shengping, L. (2012). An improved genetic algorithm for integrated process planning and scheduling. *International Journal of Advanced Manufacturing Technology*, *58*(5–8), 727–740. http://doi.org/10.1007/s00170-011-3409-0

Litao, Z., Tiejun, W., Xi, J., & Jin, J. (2012). The machine learning classifier based on Multi-Objective Genetic Algorithm. In *Proceedings - 2012 7th International Conference on Computing and Convergence Technology (ICCIT, ICEI and ICACT), ICCCT 2012* (pp. 405–409).

Liu, B. L. B., Choo, S.-H. C. S.-H., Lok, S.-L. L. S.-L., Leong, S.-M. L. S.-M., Lee, S.-C. L. S.-C., Poon, F.-P. P. F.-P., & Tan, H.-H. T. H.-H. (1994). Integrating case-based reasoning, knowledge-based approach and\nDijkstra algorithm for route finding. *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, 149–155.

Liu, Y., & Passino, K. M. (2000). Swarm Intelligence : Literature Overview. *Measurement And Control*, *2015*(614), 9. http://doi.org/http://dx.doi.org/10.1.1.135.8765

Ma, L., Wang, K., & Zhang, D. (2009). A universal texture segmentation and representation scheme based on ant colony optimization for iris image processing. *Computers & Mathematics with Applications*, *57*(11–12), 1862–1868.

http://doi.org/10.1016/j.camwa.2008.10.012

Martinoli, A., & Easton, K. (2003). Optimization of swarm robotic systems via macroscopic models. In *Proc. of the Second Int. Workshop on Multi-Robots Systems* (pp. 1–12). Retrieved from http://infoscience.epfl.ch/record/28065/files/NATO_03.pdf

Meier, A., Gonter, M., & Kruse, R. (2013). Accelerating Convergence in Cartesian Genetic Programming by Using a New Genetic Operator. *Gecco'13: Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 981–988. http://doi.org/doi:10.1145/2463372.2463481

Mitchell, M. (1995). Genetic algorithms: An overview. *Complexity*, *1*(1), 31–39. http://doi.org/10.1002/cplx.6130010108

Norouzi, M., De Bruijn, F., & Miró, J. V. (2012). Planning stable paths for urban search and rescue Robots. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7416 LNCS*, 90–101. http://doi.org/10.1007/978-3-642-32060-6_8

Noto, M., & Sato, H. (2000). A method for the shortest path search by extended Dijkstra\nalgorithm. *Smc 2000 Conference Proceedings. 2000 Ieee International Conference on Systems, Man and Cybernetics. "Cybernetics Evolving to Systems, Humans, Organizations, and Their Complex Interactions" (Cat. no.0*, *3*, 0–4. http://doi.org/10.1109/ICSMC.2000.886462

Olascuaga-Cabrera, J. G., López-Mellado, E., & Mendez-Vazquez, A. (2011). A multi-objective PSO strategy for energy-efficient ad-hoc networking. In *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics* (pp.

2632–2639). http://doi.org/10.1109/ICSMC.2011.6083994

Ouyang, X., Zhou, Y., Luo, Q., & Chen, H. (2013). A novel discrete cuckoo search algorithm for spherical traveling salesman problem. *Applied Mathematics and Information Sciences*, *7*(2), 777–784. http://doi.org/10.12785/amis/070248

Parsopoulos, K. K. E., & Vrahatis, M. N. (2011). Particle Swarm Optimization Method for Constrained Optimization Problems. *Optimization*, *181*(6), 1153–1163. http://doi.org/10.1016/j.ins.2010.11.033

Ponnambalam, S. G. (2009). Mobile robot path planning using ant colony optimization. *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 851–856. http://doi.org/10.1109/AIM.2009.5229903

Price, K., Storn, R. M., & Lampinen, J. A. (2005). Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series). *The Journal of Heredity*, *104*, 542.

Raja, P. V., & Bhaskaran, V. M. (2013). Improving the Performance of Genetic Algorithm by Reducing the Population Size. *International Journal of Emerging Technology and Advanced Engineering*, *3*(8), 86–91.

Rapaić, M. R., & Kanović, Ž. (2009). Time-varying PSO - convergence analysis, convergence-related parameterization and new parameter adjustment schemes. *Information Processing Letters*, *109*, 548–552. http://doi.org/10.1016/j.ipl.2009.01.021

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, *8*, 240–255. http://doi.org/10.1109/TEVC.2004.826071

Rimon, E., & Koditschek, D. (1992). Exact Robot Navigation Using Artificial Potential Functions. *Robotics and Automation, IEEE*, *8*(5), 501–518. http://doi.org/10.1109/70.163777

Rizzoli, A. E., Montemanni, R., Lucibello, E., & Gambardella, L. M. (2007). Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, *1*(2), 135–151. http://doi.org/10.1007/s11721-007-0005-x

Roberge, V., Tarbouchi, M., & Labonte, G. (2013). Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning. *IEEE Transactions on Industrial Informatics*, *9*(1), 132–141. http://doi.org/10.1109/TII.2012.2198665

Salama, K. M., & Freitas, A. A. (2013). Learning Bayesian network classifiers using ant colony optimization. *Swarm Intelligence*, *7*(2–3), 229–254. http://doi.org/10.1007/s11721-013-0087-6

Sa-ngawong, N., & Ngamroo, I. (2015). Intelligent photovoltaic farms for robust frequency stabilization in multi-area interconnected power system based on PSO-based optimal Sugeno fuzzy logic control. *Renewable Energy*, *74*, 555–567. http://doi.org/10.1016/j.renene.2014.08.057

Schyns, M. (2015). An ant colony system for responsive dynamic vehicle routing. *European Journal of Operational Research*, *245*(3), 704–718. http://doi.org/10.1016/j.ejor.2015.04.009

Seda, M. (2007). Roadmap methods vs. cell decomposition in robot motion planning. *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, 127–132. Retrieved from

https://planiart.usherbrooke.ca/redmine/attachments/69/Roadmap_vs_Cell_Decomposition.pdf\nfiles/1334/540-323.pdf\nhttp://www.wseas.us/e-library/conferences/2007corfu/papers/540-323.pdf

Selvi, V. (2010). Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications*, *5*(4), 1–6. http://doi.org/10.5120/908-1286

Senthil Arumugam, M., Ramana Murthy, G., Rao, M. V. C., & Loo, C. K. (2007). A novel effective particle swarm optimization like algorithm via extrapolation technique. In *2007 International Conference on Intelligent and Advanced Systems, ICIAS 2007* (pp. 516–521). http://doi.org/10.1109/ICIAS.2007.4658442

Sfeir, J., Saad, M., & Saliah-Hassane, H. (2011). An improved Artificial Potential Field approach to real-time mobile robot path planning in an unknown environment. *2011 IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, 208–213. http://doi.org/10.1109/ROSE.2011.6058518

Shi, Y., & Eberhart, R. (1998). A modified particle swarm optimizer. *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. http://doi.org/10.1109/ICEC.1998.699146

Siegwart, R., & Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. *Robotica* (Vol. 23).

Sim, K. M., & Sun, W. H. (2002). Multiple ant-colony optimization for network routing. In *Proceedings - 1st International Symposium on Cyber Worlds, CW 2002* (pp. 277–281).

http://doi.org/10.1109/CW.2002.1180890

Sirbiladze, G., & Kapanadze, M. (2012). Genetic algorithm approach for the prediction of business risks' dynamics of enterprise. In *2012 6th International Conference on Application of Information and Communication Technologies, AICT 2012 - Proceedings*. http://doi.org/10.1109/ICAICT.2012.6398507

Sivaraj, R., & Ravichandran, T. (2011). A review of selection methods in genetic algorithm. *Int J Eng Sci Tech*, *3*(5), 3792–3797.

Song, P., & Kumar, V. (2002). A potential field based approach to multi-robot manipulation. *International Conference on Robotics and Automation*, *2*(May), 1217–1222. http://doi.org/10.1109/ROBOT.2002.1014709

Storn, R., & Price, K. (1997). Differential Evolution -- A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, *11*(4), 341–359. http://doi.org/10.1023/A:1008202821328

Stützle, T., & Hoos, H. H. (2000). MAX-MIN Ant System. *Future Generation Computer Systems*, *16*(8), 889–914. http://doi.org/10.1016/S0167-739X(00)00043-1

Syed Abdullah, S. L., Hussin, N. M., Harun, H., & Abd Khalid, N. E. (2012). Comparative study of random-PSO and Linear-PSO algorithms. In *2012 International Conference on Computer and Information Science, ICCIS 2012 - A Conference of World Engineering, Science and Technology Congress, ESTCON 2012 - Conference Proceedings* (Vol. 1, pp. 409–413). http://doi.org/10.1109/ICCISci.2012.6297280

Tian, Y., Yan, L., Park, G., Yang, S., Kim, Y., Lee, S., & Lee, C. (2007). Application of RRT-

based local Path Planning Algorithm in Unknown Environment. *International Symposium on Computational Intelligence in Robotics and Automation*, 456–460.

Trelea, I. C. (2003). The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, *85*(6), 317–325. http://doi.org/10.1016/S0020-0190(02)00447-7

Tuppadung, Y., & Kurutach, W. (2011). Comparing nonlinear inertia weights and constriction factors in particle swarm optimization. *International Journal of Knowledge-Based and Intelligent Engineering Systems*, *15*(2), 65–70. http://doi.org/10.3233/KES-2010-0211

Üçoluk, G. (2002). Genetic algorithm solution of the TSP avoiding special crossover and mutation. *Intelligent Automation & Soft Computing*, *8*(3), 265–272. http://doi.org/10.1080/10798587.2000.10642829

Uguz, S., Sahin, U., & Sahin, F. (2014). Edge detection with fuzzy cellular automata transition function optimized by PSO. *Computers and Electrical Engineering*. http://doi.org/10.1016/j.compeleceng.2015.01.017

Valle, Y. Del, Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. G. (2008). Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. *IEEE Transactions on Evolutionary Computation*, *12*(2), 171–195. http://doi.org/10.1109/TEVC.2007.896686

Walton, S., Hassan, O., Morgan, K., & Brown, M. R. (2011). Modified cuckoo search: A new gradient free optimisation algorithm. *Chaos, Solitons and Fractals*, *44*(9), 710–718. http://doi.org/10.1016/j.chaos.2011.06.004

Wang, H., Yu, Y., & Yuan, Q. (2011). Application of Dijkstra algorithm in robot path-planning. In *2011 2nd International Conference on Mechanic Automation and Control Engineering, MACE 2011 - Proceedings* (pp. 1067–1069). http://doi.org/10.1109/MACE.2011.5987118

Wang, J. Q., Zhang, S. F., Chen, J., Yang, J. B., & Sun, S. D. (2010). Resource-constrained multi-project scheduling based on ant colony optimization algorithm. *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, *3*(1), 716–719. http://doi.org/10.1109/ICICISYS.2010.5658268

Wang, L., Geng, H., Liu, P., Lu, K., Kolodziej, J., Ranjan, R., & Zomaya, A. Y. (2014). Particle Swarm Optimization based dictionary learning for remote sensing big data. *Knowledge-Based Systems*, *79*, 43–50. http://doi.org/10.1016/j.knosys.2014.10.004

Wang, Y., & Chirikjian, G. S. (2000). A New Potential Field Method for Robot Path Planning. *Proceedings - IEEE International Conference on Robotics and Automation*, *2*(April), 977–982. http://doi.org/10.1109/ROBOT.2000.844727

Xu, Y., Chen, G., & Yu, J. (2006). Three Sub-Swarm Discrete Particle Swarm Optimization Algorithm. In *2006 IEEE International Conference on Information Acquisition* (pp. 1224–1228).

Yahya, B. N. ., Bae, H. ., Bae, J. ., & Kim, D. . (2012). Generating valid reference Business Process model using genetic algorithm. *International Journal of Innovative Computing, Information and Control*, *8*, 1463–1477.

Yan, X., Wu, Q., Liu, H., & Huang, W. (2013). An Improved Particle Swarm Optimization Algorithm and Its Application. *International Journal of Computer Science*, *10*(1), 316–324.

Yan Cang Li, Li Na Zhao, & Zhou, S. J. (2011). Review of Genetic Algorithm. *Advanced Materials Research*, *Volumes 17*, 365–367. http://doi.org/10.4028/www.scientific.net/AMR.179-180.365

Yang, X. S., & Deb, S. (2009). Cuckoo search via Levy flights. In *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings* (pp. 210–214). http://doi.org/10.1109/NABIC.2009.5393690

Yang, X. S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers and Operations Research*, *40*(6), 1616–1624. http://doi.org/10.1016/j.cor.2011.09.026

Yang, X. S., Deb, S., Karamanoglu, M., & He, X. (2012). Cuckoo search for business optimization applications. In *2012 National Conference on Computing and Communication Systems, NCCCS 2012 - Proceeding* (pp. 29–33). http://doi.org/10.1109/NCCCS.2012.6412973

Yang, X.-S., & Deb, S. (2010). Engineering Optimisation by Cuckoo Search. *Int. J. Mathematical Modelling and Numerical Optimisation*, *1*(4), 330–343. http://doi.org/10.1504/IJMMNO.2010.035430

Yin, C., & Wang, H. (2010). Developed Dijkstra shortest path search algorithm and simulation. In *2010 International Conference on Computer Design and Applications, ICCDA 2010* (Vol. 1). http://doi.org/10.1109/ICCDA.2010.5541129

Yu, B., Yang, Z.-Z., & Yao, B. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, *196*(1), 171–176. http://doi.org/10.1016/j.ejor.2008.02.028

Zhang, C., Zhen, Z., Wang, D., & Li, M. (2010). UAV path planning method based on ant colony optimization. *2010 Chinese Control and Decision Conference*, 3790–3792. http://doi.org/10.1109/CCDC.2010.5498477

Zhang, G., Gao, L., & Shi, Y. (2011). An effective genetic algorithm for the flexible job-shop scheduling problem. *Expert Systems with Applications*, *38*(4), 3563–3573. http://doi.org/10.1016/j.eswa.2010.08.145

Zhang, J., De-Shuang, H., & Kun-Hong, L. (2007). Multi-sub-swarm particle swarm optimization algorithm for multimodal function optimization. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on* (pp. 3215–3220).

Zhang, Y., Gong, D., & Zhang, J. (2013). Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, *103*, 172–185. http://doi.org/10.1016/j.neucom.2012.09.019

Zhang, Y., Wang, S., & Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*. http://doi.org/10.1155/2015/931256

Zhao, D., Luo, L., & Zhang, K. (2010). An improved ant colony optimization for the communication network routing problem. *Mathematical and Computer Modelling*, *52*(11–12), 1976–1981. http://doi.org/10.1016/j.mcm.2010.04.021

Zhu, H., Wang, Y., Wang, K., Chen, Y., Zhang, Y., Gong, D., … Pelevic, B. (2013). Constrained Portfolio Selection using Particle Swarm Optimization. *Expert Systems with Applications*, *8*(2), 1–14. http://doi.org/10.1016/j.ins.2012.09.030

Zhu, Y. F., & Tang, X. M. (2010). Overview of swarm intelligence. In *ICCASM 2010 - 2010 International Conference on Computer Application and System Modeling, Proceedings* (Vol. 9). http://doi.org/10.1109/ICCASM.2010.5623005

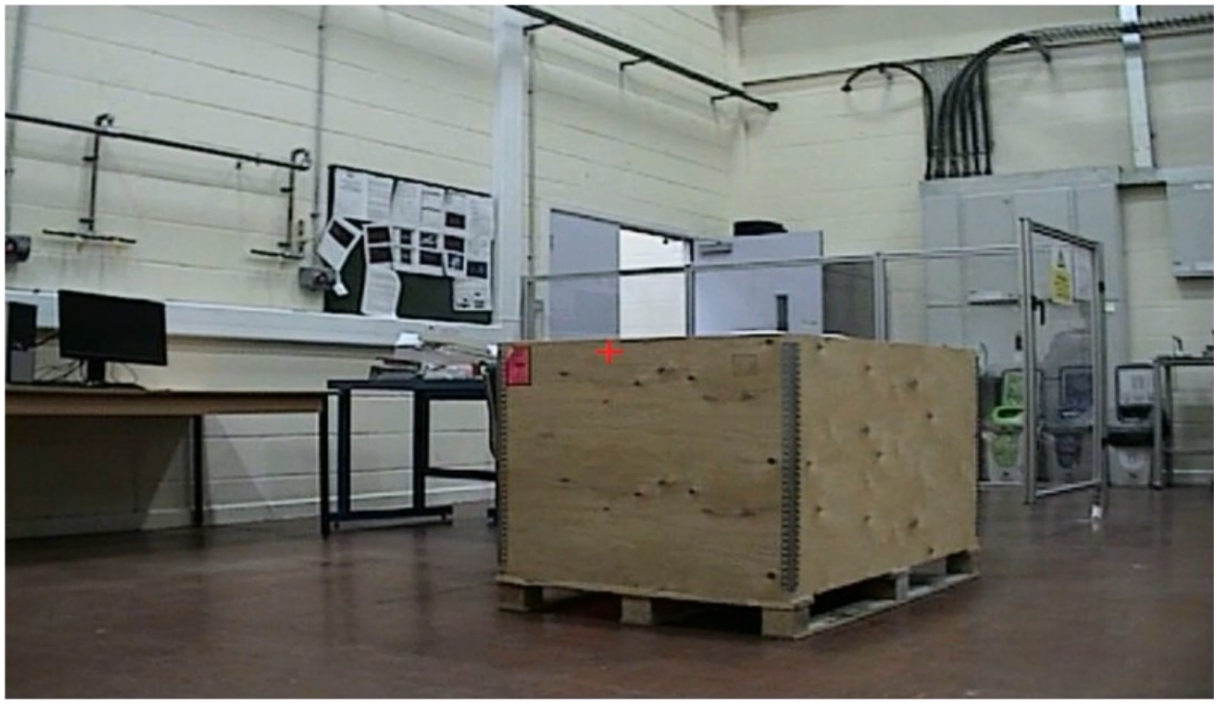
Zou, X., Ge, B., & Sun, P. (2012). Improved Genetic Algorithm for Dynamic Path Planning. *International Journal of Information and Computer Science*, *1*(2), 16–20.

APPENDICES

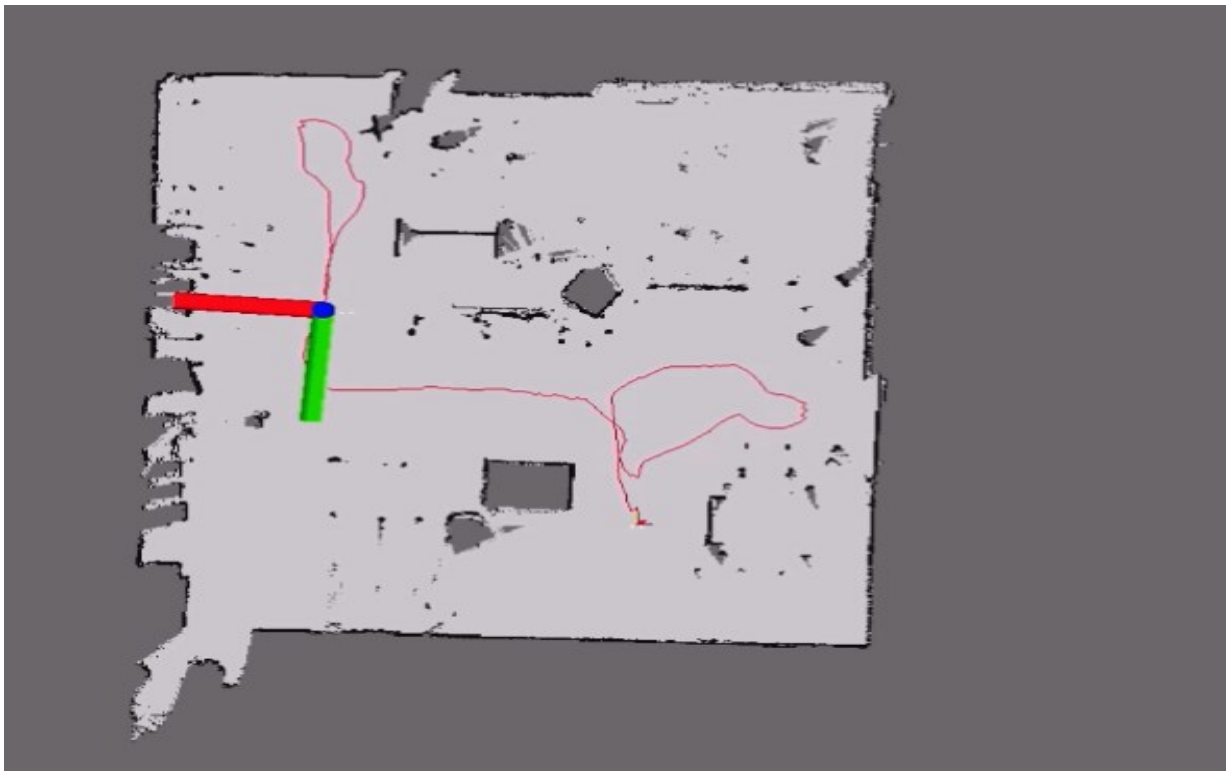
Appendix A

This appendix offers some pictures from Autonomous Indoor Mapping using Husky A200 experiments by implementing Morphology based PSO and Dynamic based PSO.

Husky A200



View from on board Wireless Camera



View of process in constructing Indoor Map

Appendix B

Finding suitable values for a_p and a_g of Morphology Particle Swarm Optimization (MPSO) with population of 10000 with 10000 iterations on over ten runs per benchmark functions.

Beale Function

Trail	**a_p and a_g = 0.5**	**a_p and a_g = 0.05**	**a_p and a_g = 0.0005**
1	1.710E-10	8.000E-12	3.000E-12
2	9.260E-09	9.900E-11	7.000E-11
3	3.000E-12	0.000E+00	1.500E-11
4	5.420E-10	0.000E+00	9.900E-11
5	6.000E-12	1.690E-10	0.000E+00
6	1.170E-09	1.865E-09	4.000E-12
7	2.000E-12	2.841E-09	3.100E-11
8	8.000E-11	1.040E-10	0.000E+00
9	4.100E-11	1.030E-09	0.000E+00
10	5.780E-10	0.000E+00	0.000E+00
Mean	1.185E-09	6.116E-10	2.220E-11
SD	2.862E-09	9.955E-10	3.493E-11

Hump Function

Trail	**a_p and a_g = 0.5**	**a_p and a_g = 0.05**	**a_p and a_g = 0.0005**
1	4.6549E-08	4.6511E-08	4.6516E-08
2	4.6786E-08	4.6519E-08	4.6515E-08
3	4.6513E-08	4.6516E-08	4.6511E-08

4	4.6524E-08	4.6538E-08	4.6512E-08
5	5.0508E-08	4.6511E-08	4.6510E-08
6	4.8527E-08	4.6511E-08	4.6520E-08
7	4.9710E-08	4.6513E-08	4.6510E-08
8	4.6516E-08	4.6511E-08	4.6513E-08
9	4.8822E-08	4.6510E-08	4.6510E-08
10	4.4797E-07	4.6523E-08	4.6510E-08
Mean	8.7843E-08	4.6516E-08	**4.6513E-08**
SD	1.2654E-07	8.7312E-12	**3.3682E-12**

Matyas Function

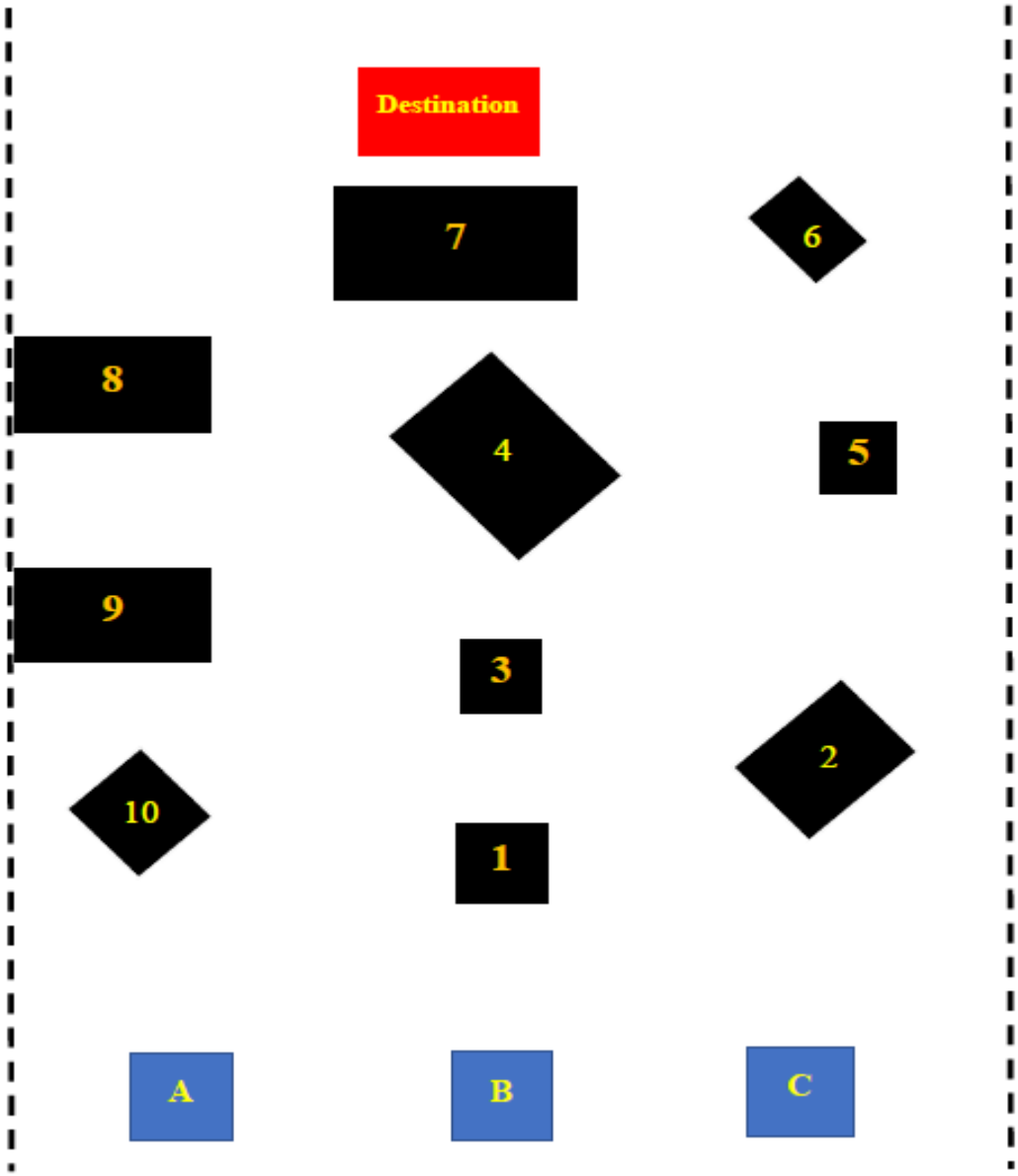
Trail	a_p **and** a_g **= 0.5**	a_p **and** a_g **= 0.05**	a_p **and** a_g **= 0.0005**
1	2.000E-12	2.000E-11	1.220E-10
2	1.912E-09	0.000E+00	0.000E+00
3	1.670E-10	1.470E-10	0.000E+00
4	4.650E-10	0.000E+00	0.000E+00
5	2.800E-11	1.560E-10	2.020E-10
6	5.000E-12	0.000E+00	0.000E+00
7	3.564E-08	0.000E+00	5.500E-11
8	3.580E-10	3.700E-11	5.000E-12
9	6.200E-11	0.000E+00	0.000E+00
10	2.380E-10	4.738E-09	0.000E+00
Mean	3.888E-09	5.098E-10	**3.840E-11**
SD	1.117E-08	1.487E-09	**6.991E-11**

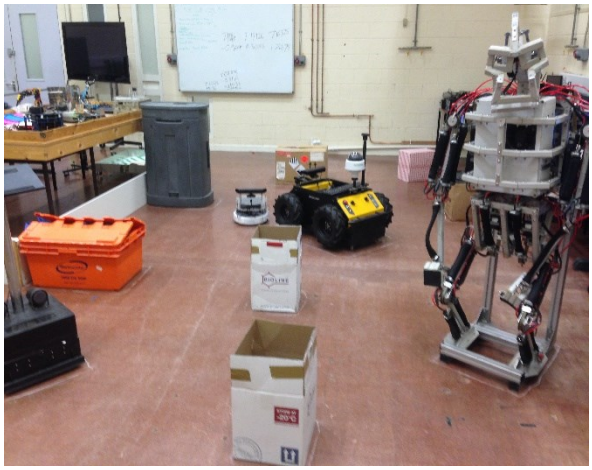
Rastrigin Function

Trail	a_p and a_g = 0.5	a_p and a_g = 0.05	a_p and a_g = 0.0005
1	1.993E-05	5.903E-05	2.600E-11
2	2.997E-04	3.186E-06	2.020E-10
3	4.500E-11	3.113E-06	5.030E-10
4	8.900E-06	1.794E-07	3.158E-08
5	8.689E-07	7.400E-11	8.600E-11
6	1.851E-03	1.987E-09	4.300E-11
7	2.735E-06	2.660E-10	5.983E-08
8	3.901E-05	7.000E-12	3.564E-09
9	1.064E-04	9.841E-07	2.484E-05
10	7.510E-08	1.241E-08	3.432E-09
Mean	2.329E-04	6.651E-06	**2.494E-06**
SD	5.762E-04	1.845E-05	**7.853E-06**

Appendix C

Testing environment with a combination of odd and regular shape of obstacles.






The experiments have been carried indoor. Appendix C shows environment layout. Black blocks represent obstacles with blue point represents the starting point and red point represents the goal points. The shortest distance between them in straight line is 6.041. There are 11 obstacles in the map with smallest size is 72cm^2 (obstacle no 9 and 11) and largest size is 63504cm^2 (obstacle no 5). The largest obstacle gap is between 10 and 11 where the distance is 140cm. The smallest obstacle gap is between obstacle 8 and 10 with 50cm follow by obstacle 7 and 9 with 55cm.