SEED DISPERSER ANT ALGORITHM FOR OPTIMIZATION

CHANG WEN LIANG

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

SEED DISPERSER ANT ALGORITHM FOR OPTIMIZATION

CHANG WEN LIANG

DISSERTATION SUBMITTED IN FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ENGINEERING SCIENCE

FACULTY OF ENGINEERING
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate:	Chang Wen Liang KGA
Matric No:	
Name of Degree:	Master of Engineering Science
Title of Dissertation:	SEED DISPERSER ANT ALGORITHM FOR OPTIMIZATION
Field of Study:	Algorithms and Optimizations

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature   Date:

Subscribed and solemnly declared before,

Witness's Signature   Date:

Name:

Designation:

SEED DISPERSER ANT ALGORITHM FOR OPTIMIZATION

ABSTRACT

The Seed Disperser Ant Algorithm (SDAA) is developed based on the evolution or expansion process of Seed Disperser Ant (*Aphaenogaster senilis*) colony. The genotype of every ant is represented in binary form as the variables. These binary variables are used to locally search for optimum solution. SDAA is developed using the concept of male ants performing nuptial flights to generate new superior colonies. The new colonies produce better male ants that repeat the nuptial flight cycle in following generation. New young queens are produced by the colony that migrates to establish new colonies after local optimum solution reached to start new local search. Nuptial flight and new young queens' production aid in enhanced search exploitation and exploration respectively. This diversifies the search for global optimum. The classical benchmark problems and composite benchmark functions from Congress on Evolutionary Computation (CEC) 2005 special session is used for validate SDAA. Engineering optimization has become important in design problems to reduce error and faulty production as many constrained condition should be taken in to account before manufacturing. Also, data clustering has become popular in data mining in recent time due to data explosion. In this research, we applied SDAA to solve the constrained engineering problems and introduce an efficient data clustering algorithm which is hybrid of K-means and SDAA. The optimal results obtained for constrained engineering problems as well as data clustering are very promising in terms of quality of solutions and convergence speed of the algorithm.

Keywords: Optimization, composite benchmark function, constrained engineering problem, data clustering, Seed Disperser Ant Algorithm.

SEED DISPERSER ANT ALGORITHM UNTUK PENGOPTIMUMAN

ABSTRAK

Seed Disperser Ant Algorithm (SDAA) adalah inspirasi daripada proses evolusi koloni semut penyebar biji benih iaitu spesies *Aphaenogaster senilis*. Genotip semut sebagai pembolehubah diwakili dalam bentuk nombor perduaan. Pembolehubah ini digunakan untuk pencarian sebahagian mencapai peyelesaian optimum. SDAA merupakan algoritma berdasarkan penerbangan mengawan semut jantan untuk menghasilkan generasi dan koloni baru yang lebih unggul. Semut jantan yang telah bertambah baik dalam generasi atau koloni baru akan berterusan melaksanakan penerbangan mengawan dan proses ini berulang untuk menghasilkan generasi atau koloni baru yang bertambah baik. Sekiranya koloni telah mencapai satu tahap kematangan, dengan kata lain, optimum sebahagian yang dicari telah capai, ratu semut muda akan dilahirkan oleh koloni dan berhijrah untuk menubuhkan koloni baru. Proses penebangan mengawan dan kelahiran ratu semut muda masing-masing membantu mempertingkatkan eksploitasi carian dan penerokaan. Ini memperluaskan pencarian optimum secara global. SDAA disahkan dengan menyelesaikan masalah ukur rujuk klasik dan masalah ukur rujuk komposit dari Congress on Evolutionary Computation (CEC) 2005 sesi khas. Pengoptimuman kejuruteraan telah menjadi unsur penting dalam masalah reka bentuk untuk mengurangkan kesilapan dan kerosakan produksi kerana banyak syarat dikekang perlu diambil kira sebelum pembuatan. Selain itu, pengelompokan data telah menjadi popular dalam bidang perlombongan data disebabkan data dihasilkan secara lampau kebelakangan ini. SDAA digunakan untuk menyelesaikan masalah kejuruteraan dikekang dan algoritma data pengelompokan yang cekap diperkenalkan dalam penyelidikan ini iaitu hibrid K-means dan SDAA. Keputusan optimum diperolehi bagi masalah kejuruteraan dikekang serta pengelompokan data adalah sangat memberangsangkan dari segi kualiti penyelesaian dan kelajuan penumpuan algoritma.

Kata kunci: Pengoptimuman, fungsi penanda aras komposit, masalah kejuruteraan dikekang, clustering data, Seed Disperser Ant Algorithm.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my supervisor also as my advisor, Associate Professor Dr. Jeevan Kanesan and my second supervisor Associate Professor Ir. Dr. Harikrishnan A/l Ramiah for the continuous support of my Master study and related research. Their guidance helped me in all the time of research and preparing of this dissertation. Apart from the technical knowledge, I have learned so much from their insightful technical writing skill.

Next, my gratitude goes to Assistant Professor Dr Anand J Kulkarni from Symbiosis Institute of Technology, India. He had spent his time and effort on guiding me for writhing algorithm code effectively when he visiting Malaysia. When I am lost on my research, he had help in checking my algorithm's concept and guides me back to my research objective. He also shared his experiences to help me in my research. I am also thanks for his advice of the conference and journal paper writing.

I thank my fellow lab mates Mr. Tey Yong Yuen, Mr. Chong Wei Keat, Ms. Nandini Vitee, Mr. Lim Chee Cheow, Mr. Jalil Ahmadian, Ms. Teo Ting Huan, and Mr. Mohammad Zihin bin Mohd Zain. They have accompanied me in the same lab for doing research, especially Mr. Chong Wei Keat, who has provided technical support for the lab's computers and printers and Mr. Tey Yong Yuen, who always share postgraduate's information and updates. With them, I am not lonely in doing my research, able to ask and refer opinion from them.

The financial support of Universiti Malaya High Impact Research Grant UM.C/HIR/MOHE/ENG/51 from the Ministry of Higher Education Malaysia is gratefully acknowledged.

Last but not the least, I would like to thank my parents and to my sisters for supporting me spiritually throughout my research of my Master study and my life in general.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

SDAA	:	Seed Disperser Ant Algorithm
CEC	:	Congress on Evolutionary Computation
EC	:	evolutionary computation
SI	:	Swarm Intelligence
NFL	:	No Free Lunch theorem
MSDAA	:	modified Seed Disperser Ant Algorithm
EAs	:	Evolutionary Algorithms
GA	:	Genetic Algorithms
DE	:	Differential Evolution
MA	:	Memetic Algorithms
CMA-ES	:	Covariance Matrix Adaptation Evolution Strategy
PSO	:	Particle Swarm Optimization
CS	:	Cuckoo Search
GWO	:	Grey Wolf Optimizer
EPSO	:	Evolutionary Particle Swarm Optimization
IPSO	:	Iteration particle swarm optimization
GPSO	:	Global Particle Swarm Optimization
CPSO	:	Chaos-Particle Swarm Optimization
ES	:	Evolution Strategies
ACO	:	Ant Colony Optimization
HBMO	:	Honey-Bee Mating Optimization
CI	:	Cohort Intelligence
CMIWOKM	:	hybrid Cloud Model invasive weed optimization with K-Means
SRPSO	:	stochastic ranking particle swarm optimization

MSBA	:	Mutable Smart Bee Algorithm
EIWO	:	Enhanced Invasive Weed Optimization
RHPSO	:	Robust Hybrid Particle Swarm Optimization
BA-DE	:	Hybrid Bat-Inspired Algorithm with Differential Evolution
SA	:	Simulated Annealing
TS	:	Tabu Search
LSB	:	least significant bit
MSB	:	most significant bit
FEs	:	function evaluations
Std.	:	standard deviation
CF	:	Composite Benchmark Function
ASCHEA	:	Adaptive Segregational Constraint Handling Evolutionary Algorithm
DELC	:	Differential Evolution with Level Comparison
HPSO	:	Hybrid Particle Swarm Optimization
CDE	:	Co-evolutionary Differential Evolution
CAEP	:	cultural algorithms with evolutionary programming
NM-PSO	:	Hybrid Nelder-Mead simplex search & Particle Swarm Optimization
ABC	:	Artificial Bee Colony Algorithm
Q_a	:	Queen a
Q_b	:	Queen b
X^1	:	Male ant 1
X^A	:	Male ant A
Q_1	:	Queen 1
Q_n	:	Queen n
$f(L_h)$	:	Function to be optimize
l_j	:	Single input for function $f(L_h)$, where $l_j = x_j$

Ψ_j^{lower}	:	Original lower boundary of input l_j
Ψ_j^{upper}	:	Original upper boundary of input l_j
L_h	:	Input for function $f(L_h)$
$g_k(L_h)$	:	Constrained condition function
R_h	:	Refer gene, colony identical gene carried by Queen
C_i	:	ith colony
Q_d	:	Mated queen carried diploid gene
G	:	Number of iterations
C	:	Number of colonies
r	:	Shrinking factor
ε	:	Convergence parameter
S	:	Saturation number
LB_j	:	Search space lower boundary of input l_j
UB_j	:	Search space upper boundary of input l_j
N	:	Number of dimension
$L_{h(fit)}$	:	Fittest input L_h
$L_{h(best)}$	:	Current best solution input found
B_j	:	Search space boundary size
D	:	Clustering data set
M	:	Number of data objects
K	:	Number of clusters
Y_i	:	ith object in data set
X_i	:	Centroid of the ith cluster
$F(X,Y)$	:	Clustering Function to be optimize

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Motivation

The behaviour modelling of social insects as problem solving technique for efficient search has been the main context in swarm intelligence field. Various evolutionary computation (EC) techniques have been introduced in the field of optimization. This includes Swarm Intelligence (SI) inspired from social behaviours of insects and animals as well as Swarm inspired meta-heuristics that applied to many kind of optimization problems, for example: constrained engineering problems, combinatorial problems including traveling salesman problem, global optimization, vehicle routing problems, control engineering, assignment problems, traffic system design, scheduling, data clustering, etc.

The year 2012 report shows that there were 2.5 Exabyte (2.5 billion gigabytes) of data being generated daily. The data sets are large and complex that need tools and approaches to process them. Thus, data processing is the fundamental and challenge for a data analyst. Data clustering is the most common technique used as unsupervised classification technique in data processing. A clustering technique divides the data groups into different cluster by bunch up the data with equal or similar characteristics or pattern. Generally, there are few major purposes of using data clustering which include gain insight into data, generate hypotheses, detect anomalies, and identify salient features, identify the degree of similarity among forms or organisms, and also organize the data and summarizing it through cluster prototypes.

Many industrial engineering activities associate with high complexity and unstructured conditions. These real-life problems are difficult to model because of require unique factors. For constrained engineering problems, swarm intelligence techniques give potential breakthrough. They are able to provide near-optimal or optimal solutions, thus enabling to choose the best solution based on the criteria that the industry needs.

1.2 Challenge

The No Free Lunch (NFL) theorem (Wolpert & Macready, 1997) has proved a particular meta-heuristic may obtain very convincing results on a set of problems, but also may show poor performance on a different set of problems. The results also show the importance of incorporating problem-specific knowledge into the behaviours of the algorithm used. This makes the field of optimization based meta-heuristics very active in enhancing current approaches and proposing new meta-heuristics every year. This motivates a new meta-heuristic to be developed with inspiration from seed disperser ant evolution concept for solving clustering problems. Also the evolutionary based algorithm developed in this research showing the reduction of tuning variables and parameter by modifying the algorithm to merge with others algorithm.

1.3 Design and Solution

In this research, Seed Dispenser Ant Algorithm (SDAA) is developed and verified with several classical and composite benchmarks. Seed Dispenser Ant Algorithm (SDAA) (Chang, Kanesan, & Kulkarni, 2015) is inspired from *Aphaenogaster senilis* (Cheron, Doums, Federici, & Monnin, 2009) evolution process. The queen ant/s with diploid genes and male ants provide haploid genes are represented using binary code number. Nuptial flight (Kenne & Dejean, 1998) occurred when male ants fly out to mate with the queen ant of others colony. The mated queens carrying the male haploid then will produce offspring for the next generation. The optimum solution search is done using both local and global search features. Each offspring is produced using male ant haploid gene copies their alleles from the adjoining haploid refer gene which is haploid of queen’s diploid. These offspring have high similarity to each other that useful in local search exploitation.

Once the offspring production search process saturated, the chosen colony will produce fittest young queen to form new colony. This is used for exploratory search to avoid trapped in local optima. The chosen young queen will create a new colony and continue with exploitation local search process. The exploitation and exploration search process is repeated and converges to achieve optimal solution.

1.4 Objectives

The main objective of this thesis is to develop a new algorithm which able to solve several kind and fields of problems using the concept which can observe from nature. Many nature inspired algorithms created for solving certain type of problem, this research is done for developing a new algorithm which able to solve as many problems as it could. Thus, the objectives under scrutiny are:

- To create and develop a new algorithm inspired from nature.
- To maximize the variety of problems solved using the same algorithm.
- To obtain more accuracy and persistency of the solution.

1.5 Contributions

In this work, we are proposing its ability to handle constrained optimization and data clustering. Now days, there are many real world engineering problems such as engineering design problems requiring optimization to reduce cost and save energy, improve safety, quality of life, and efficiency of work. For an engineering design problem, there are many constrained conditions that should be taken into account before manufacturing the design.

SDAA has advantages of on its searching technique where it searches by binary bit changing process to generate new solutions. The offspring generation advocates search exploitation where as young queen generation explores for better solution. Hence, SDAA aggressively searches optimum solutions within the search domain and this helps to escape local optima.

Also, there is a motivation for improvement on SDAA for data clustering problems, where function evaluations could be reduced significantly by some modification. In SDAA, the search exploration cost huge function evaluations. This dooms SDAA to the limited application as problems such as data clustering requires rapid solution. Therefore the young queen generation responsible for search exploration is removed in the modified SDAA (MSDAA). Besides, the initial search in SDAA is simply based on random generation of solutions. However, K-means in MSDAA is used to generate initial solutions. Due to the simplicity and ease of application, K-means is adopted for solving data clustering problems. Though K-means has a tendency to trap easily in local optima, the offspring generation feature of SDAA ensures the solution escapes from the local optima. The disadvantages of both SDAA and K-means are solved by hybridizing both techniques in MSDAA. A modified SDAA (MSDAA) is proposed which hybridizes the advantages of both SDAA and K-means for solving data clustering problems. The purpose of developing MSDAA is to enhance the original SDAA to produce improved accuracy, lower standard deviation with lower function evaluations and reduce the tuning parameters of SDAA with the aid of K-means in the algorithm.

1.6 Dissertation Outline

The organizations of this paper are Section 1 to Section 5. Introduction in Section 1 contain motivation to start this research work, problems or challenges faced, the objectives of the research, solution and contribution of this work. Next, continue with literature review in section 2. This Section review the previous algorithms used in optimization field and types of problem solved. In this Section, the past and recent ideas and methods used is referred and compared to create new algorithm. The original creation of SDAA and the modified SDAA for clustering is described and explained in Section 3. This Section explains the details and the design of the algorithms. Then followed by Section 4 for the Results and discussion, and lastly Section 5 provides the conclusions of this research.

CHAPTER 2: TYPES OF COMPUTATIONAL INTELLIGENCE

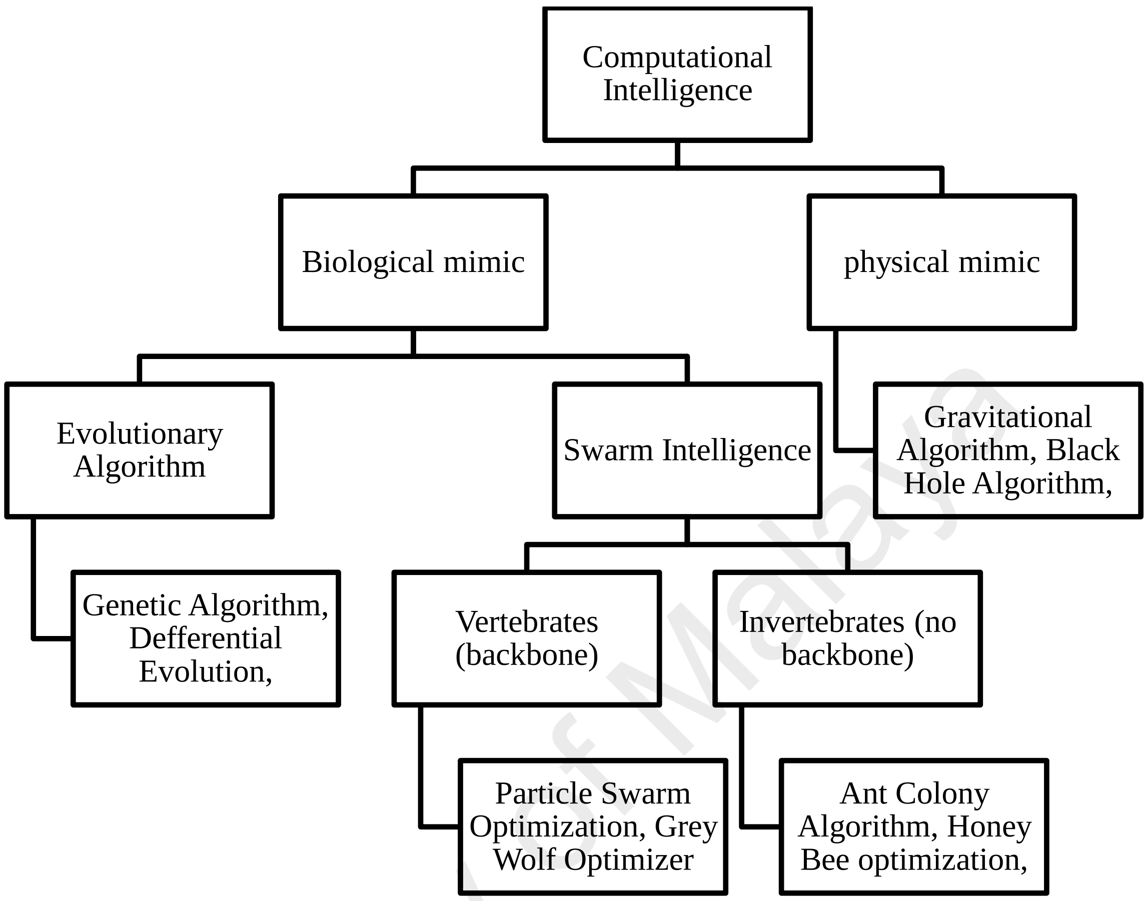
Computational Intelligence
Biological mimic
physical mimic
Evolutionary Algorithm
Swarm Intelligence
Gravitational Algorithm, Black Hole Algorithm,
Genetic Algorithm, Defferential Evolution,
Vertebrates (backbone)
Invertebrates (no backbone)
Particle Swarm Optimization, Grey Wolf Optimizer
Ant Colony Algorithm, Honey Bee optimization,

Figure 2.1: Computational intelligence categories

The computational intelligence algorithms categorised into physical and biological mimic. Physical mimic algorithms are developed based on nature phenomena, existing physic or mathematical formulas. For example, Gravitational Algorithm is developed based on Newton's law of universal gravitation; Black Hole Algorithm is developed based on black hole phenomena. Biological mimics algorithms can be separated into two groups which are Evolutionary Algorithms and Swarm Intelligence as shown in Figure 2.1. The Evolutionary Algorithms are developed based on the surviving and evolving ability of selected living creature for enhancing fitness for continuation of the next generation. These algorithms mostly use genetic code representation for improving the fitness. Examples of Evolutionary Algorithms are Genetic Algorithm and Differential Evolution. For the category of Swarm intelligence, we can divide it into 2 types of swarms mimic, which is vertebrate and invertebrate animals. Most of the Swarm Intelligence algorithms mimic the food foraging ability or hunting strategy of the swarm. Vertebrate animals' category algorithms have Grey Wolf Optimizer which mimics the wolf hunting strategy, and Particle Swarm Optimization mimics the swarm of fish or birds. On the other hand, for invertebrate animals' category algorithms, we have Ant Colony Algorithm and Honey Bee optimization which mimic the food foraging ability.

2.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) is a category of algorithm created based on evolution and survival of the fittest individual. EAs use genetic operators to exploit population evolution for global optimum search such as Genetic Algorithms (GA) (Goldberg, Zakrzewski, Chang, & Gallego, 1997; Mitchell, 1998), Differential Evolution (DE) (Qin, Huang, & Suganthan, 2009; Storn & Price, 1997) and Memetic Algorithms (MA) (Moscato, 1989; Moscato, Cotta, & Mendes, 2004). MA embeds individual learning procedure proficient of performing local search refinements through genetic operators which can categorize as an alternative of Genetic Algorithm (GA). In addition, DE has been emphasizing on the mutation that efficiently using exploratory search for global optimum solutions. Besides, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Auger & Hansen, 2005) also categories into EAs. It indicates a mutation as an evolution strategy that adapts the full covariance matrix of a normal search distribution. The CMA-ES has an important property which is its constant against linear transformations of the search space compared to many other Evolutionary Algorithms.

2.2 Swarm Intelligence

Swarm Intelligence (SI) is a category of algorithm simulates the activities and movement of insects or animals. There are many SI techniques such as particle swarm optimization (PSO) (Eberhart & Kennedy, 1995), Cuckoo Search (CS) (X.-S. Yang &

Deb, 2009) and Grey Wolf Optimizer (GWO) (Mirjalili, Mirjalili, & Lewis, 2014). PSO is developed by mimicking the crowd of animals' movement as in a flock of birds or a school of fish. The concept used to develop PSO is based on collision-proof birds' movement. By following the creation of PSO, many variations or enhance PSO were developed such as Evolutionary Particle Swarm Optimization (EPSO) (Miranda & Fonseca, 2002), Iteration particle swarm optimization (IPSO) (Lee & Chen, 2007), Global Particle Swarm Optimization (GPSO) (Jamian, Abdullah, Mokhlis, Mustafa, & Bakar, 2014) and Chaos-Particle Swarm Optimization (CPSO) (L. Liu, Zhong, & Qian, 2010). EPSO is a general-purpose algorithm that emphasize in Evolution Strategies (ES) using PSO concept. In ES, a number of models have been developed that rely on Darwinist selection to promote progress towards the optimum. IPSO is developed by sizing the distributed generation unit and applying new velocity of each particle before updating the position. On the other hand, GPSO share information about the particle position between the dimensions at any iteration to fasten the convergence process of classical PSO. CPSO is developed by combining chaos search strategy with PSO. Chaos is deterministic and it can be generated using fixed rules or equations. The "butterfly effect" in chaos can guide to a very different result in a very small change. CS is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of different species birds as host birds. Studies show that CS idealized such breeding behaviour that it could outperform existing algorithms such as PSO. Besides, GWO was inspired from grey wolves' (*Canis lupus*) leadership hierarchy and hunting mechanism.

There are many algorithms developed for solving these optimization problems such as Particle swarm Optimization (PSO) (Tsai & Kao, 2011), Ant Colony Optimization (ACO) (Shelokar, Jayaraman, & Kulkarni, 2004), Honey-Bee Mating Optimization (HBMO) (Fathian, Amiri, & Maroosi, 2007), Cohort Intelligence (CI) (Kulkarni, Durugkar, & Kumar, 2013), hybrid Cloud Model Invasive Weed Optimization with K-Means

(CMIWOKM) (Pan, Li, Ouyang, Zhou, & Xu, 2014), stochastic ranking particle swarm optimization (SRPSO)(Ali, Sabat, & Udgata, 2012), mutable smart bee algorithm (MSBA) (Mozaffari, Gorji-Bandpy, & Gorji, 2012), enhanced invasive weed optimization (EIWO) algorithm (Ramezani, Ahangaran, & Yang, 2013), Robust Hybrid Particle Swarm Optimization (RHPSO) (Xu, Geng, Zhu, & Gu, 2013), and Hybrid Bat-Inspired Algorithm with Differential Evolution (BA-DE) (Pei, Ouyang, & Tong, 2012).

2.3 Exploration and Exploitation

Most of the computational intelligence algorithms have two common aspects, which are exploration and exploitation search. The exploration is the ability of expanding search space, where the exploitation is the ability of finding the optima around a good solution. The exploratory search in an algorithm explores the search space to find new solutions to avoid trapping in a local optimum. Then the algorithm uses exploitation search to tunes it iteration by iteration to achieve an optimum solution. The exploration and exploitation search of different algorithms used different approaches and operators. These cause the different efficiency of different algorithms.

2.4 Algorithms / Methods comparison

None of the algorithms is perfect that able to solve all kind of problems. Every algorithm has its own strength and also weakness. Several algorithms are studies and compared. The algorithms include K-means, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant colony Optimization (ACO), Differential Evolution (DE), Gravitational Search Algorithms (GSA), Grey Wolf Optimization (GWO), Covariance Matrix Adaptation Evolution Strategy (CMA-ES), and Cuckoo Search (CS). By studying the algorithms, based on the behaviour and test, advantages and disadvantages are compared as the table below:

Table 2.1: Advantages and Disadvantages of Algorithms

Algorithm	Advantages	Disadvantages
K-Means	Fast, robust and easier to understand.	Easily trap in local search. Fails for non-linear data set. Unable to handle noisy data and outliers
ACO	Inherent parallelism. Efficient for Traveling Salesman Problem and similar problems. Can be used in dynamic applications.	Sequences of random decisions (not independent). Probability distribution changes by iteration. Time to convergence uncertain.
PSO	No overlapping and mutation calculation. Calculation is very simple.	Easy to fall into local optimum in high-dimensional problem. Low convergence rate in the iterative process.
GA	Parallelizability. Support multi-objective optimization. Good in noisy environment search.	Slow convergence, time-consuming. Solution depending on design of objective function representation.
DE	Support non-differentiable, nonlinear and multimodal cost functions. Parallelizability. Consistent convergence to the global minimum.	Parameter tuning of necessary. Same parameters may not guarantee the global optimum solution.
GSA	Flexible. Balance exploration and exploitation search. Support nonlinear problem, multi-objective problem.	Each iteration needs too many computations.
GWO	Only 2 main parameters to be adjust.	Low precision. Weak in local search.
CMA-ES	Support high dimensions problems. Large search interval.	Weak in solving partly separable problems, small dimension problems. High function evaluations.
CS	Easier to application and fewer tuning parameters.	Easy to fall into local optimal.

2.5 Data Clustering and Real World Engineering Problems

The technique to process big data has become a fundamental and critical challenge for modern society. Most of the data is stored digitally in electronic media, thus providing huge potential for the development of automatic data analysis, clustering, and retrieval techniques (Hashem et al., 2015; Jain, 2010; C. Yang et al., 2014). Data clustering approach had been applied to variety of applications, such as code book generation, data mining, image segmentation, pattern recognition, and sensor clustering. There are many algorithm developed which able to solve data clustering problems, for example several Swarm Intelligence (SI) (Blum & Li, 2008) methods and Evolutionary Algorithms (EAs) (Back, 1996) also have been used to solve data clustering problems. The most well known SI and EA methods implemented for data clustering are Particle swarm optimization (PSO) (James Kennedy, 2010; J. Kennedy & Eberhart, 1995; Tsai & Kao, 2011) and Genetic Algorithm (GA) (Goldberg et al., 1997; Maulik & Bandyopadhyay, 2000; Mitchell, 1998) respectively. Honeybee-mating optimization (HBMO)(Fathian & Amiri, 2008) and Ant Colony Optimization (ACO) (Dorigo & Birattari, 2010) are both SI methods created for solving data clustering problems. Simulated Annealing (SA) (Niknam & Amiri, 2010; Selim & Alsultan, 1991) and Tabu Search (TS) (Al-Sultan, 1995; Niknam & Amiri, 2010) are non SI/EA methods used to solve data clustering problems. Other than SI and EAs methods, there is also learning based algorithm such as Cohort Intelligence (CI) (Krishnasamy, Kulkarni, & Paramesran, 2014) developed for optimization in clustering.

Real world engineering problems increases day by day. This is due to new development and improvement of technology now days. Whenever there is an improvement, there must be some problem to be solved. There are several algorithms were developed to solve constrained problems or engineering problems, for example: mutable smart bee algorithm (MSBA) (Mozaffari et al., 2012), Robust Hybrid Particle

Swarm Optimization (RHPSO) (Xu et al., 2013), and Hybrid Bat-Inspired Algorithm with Differential Evolution (BA-DE) (Pei et al., 2012), stochastic ranking particle swarm optimization (SRPSO)(Ali et al., 2012), mutable smart bee algorithm (MSBA) (Mozaffari et al., 2012), enhanced invasive weed optimization (EIWO) algorithm (Ramezani et al., 2013), Robust Hybrid Particle Swarm Optimization (RHPSO) (Xu et al., 2013), and etc.

CHAPTER 3: SOLUTION/ METHODOLOGY

Seed Disperser Ant Algorithm (SDAA) (Chang, Kanesan, & Kulkarni, 2015) is a new meta-heuristic inspired from nature. SDAA is proposed based on seed disperser ant species name *Aphaenogaster senilis* (Cheron et al., 2009) evolution process. SDAA mimics evolutionary strategy of seed disperser ant to enhance its fitness of future generation to continue survives. Common evolutionary techniques concern in genetic processes. In SDAA, ants' haploid-diploid genetic idea invented by uses of Kin Altruism (Ashton, Paunonen, Helmes, & Jackson, 1998; Osiński, 2009) for its evolution of the colony. The haploid-diploid genetic code in binary form is represented the available solution of the problem. The infertile female workers and queen ant/s which have diploid genes are populated in the colony. Haploid genes provided by male ant used in performing nuptial flight (Kenne & Dejean, 1998). Male ants fly out in nuptial flight process and mate with the queen from others colony. The mated queens will generate offspring in their respective colony. The optimum search of SDAA is established in locally and also globally. The offspring is generated when male provided haploid gene replicate alleles from the contiguous female's haploid gene. This offspring production performs as local search and the offspring produced are highly related to each other for search exploitation. A fit young queen is produced from the chosen fittest colony when saturated on local search. The fit young queen will then migrate and establish its own new colony. This search helps to explore the global optimum solution. The fit young queen established its new colony will start the process of local search. All there process will continuous repeat again until saturation or limit of iteration. Both local and global search process continuous converges to the optimum solutions.

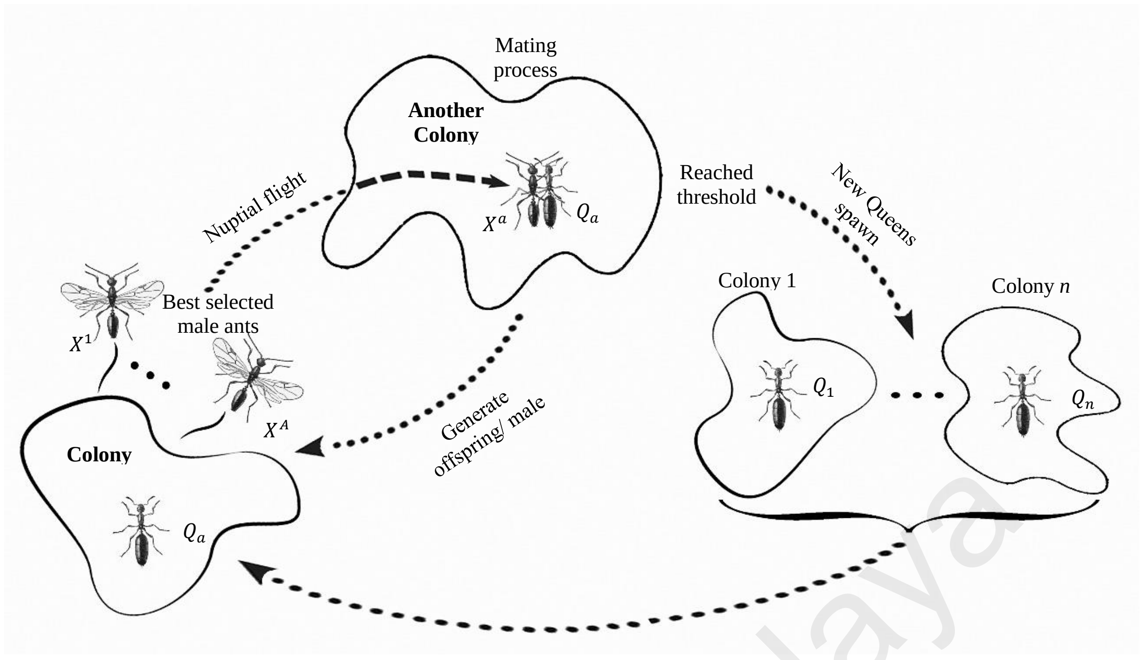

Figure 3.1: Evolution concept of SDAA

In Figure 3.1, the evolution concept of SDAA shows that from a colony, the queen Q_a generate offspring which are X^1 to X^A. The best male ant selected to perform nuptial flight for mating with queen Q_b from another colony. This process continues with all colonies until threshold reached. Where there is no improvement in generating offspring. Next, the best colonies will spawn new queens Q_1 to Q_n and become new colonies. SDAA perform the search in binary form, where it depends on the precisions used for the input variables. The precisions of the input variables affect the bits number used for binary representation.

The general minimization of unconstrained problem as shown as below:

$$\textbf{Minimize } \boldsymbol{f(L_h) = f(l_1, \dots\ l_j, \dots\ l_N)} \quad \textbf{(3.1)}$$

Subject to

$$\boldsymbol{\Psi_j^{lower} \leq l_j \leq \Psi_j^{upper}\ , j = 1, \dots N} \quad \textbf{(3.2)}$$

There are several classical constrained engineering problems difficult to be optimized. These classical constrained engineering problems include single-objective test problem (Floudas & Pardalos, 1990), tension spring design problem (Belegundu & Arora, 1985), pressure vessel design problem (Kannan & Kramer, 1994) and welded beam design problem (Coello Coello, 2000) as shown in Appendix A. These problems have constrained conditions where all the design variables interrelated with each other. To obtain the minimum result, the design variables must satisfy all the constrained conditions. This makes optimizing constrained engineering problems cumbersome.

The objective function of constrained engineering problem is embedded with the constrained condition. The input variables x will evaluated by the constrained condition's functions $g_i(x)$. If the input variables x satisfied the conditions, then the input variables x will be evaluated by the objective function $f(x)$, else the $f(x)$ set to be equal to infinite. The N number of variables x in SDAA will represented by L_h:

$$\boldsymbol{L_h = [x_1, \dots x_j, \dots x_N]\,,\ j = 1, \dots N} \qquad \textbf{(3.3)}$$

The objective function $f(x)$ for SDAA solving constrained engineering problem is shown as below:

$$\boldsymbol{f(x) = \min f(L_h)} \qquad \textbf{(3.4)}$$

Subject to:

$$\boldsymbol{g_k(L_h) \leq 0} \qquad \textbf{(3.5)}$$

Objective function $f(x)$ assumed to be infinity if any requirement of constrained condition's functions $g_k(x)$ is not satisfied.

3.1 Seed Disperser Ant Algorithm

Considering the fitness of male ant as the objective function $f(L_h)$ in a colony. The gene code of male ant is represented by $L_h = (l_1, \dots l_j, \dots l_N)$. Every colony's queen is assumed to give birth of many new young queens and with the identity gene of the colony represented by the refer gene $[R_h]_{C_i}$. A complete diploid gene is represented by combine a pair of haploid genes, which from the binary gene code of male ant and its complement $[R_h]_{C_i}$ as the mated queen $[Q_d]_{C_i}$. The equation representation is shown in equation (3.6).

$$[\boldsymbol{Q_d}]_{C_i} = [\boldsymbol{L_h}]_{C_i}[\boldsymbol{R_h}]_{C_i} \quad \textbf{(3.6)}$$

Where,

$$\boldsymbol{C_i} = \boldsymbol{i}^{\text{th}} \text{ colony} \quad \textbf{(3.7)}$$

The SDAA initialize with tuning parameters such as number of iterations G, shrinking factor r, number of colony C, saturation number S, convergence parameter ε, and number of decimal points for input variables' precisions. Upper boundary UB_j and lower boundary LB_j are fixed at the beginning as given in equations below. Number of bits in binary form is determined by the precisions in number of decimal points for input variables. The input variables (L_h) in this research used precisions up to 5 decimal points.

$$\boldsymbol{UB_j} = \boldsymbol{\Psi_j^{upper}} \quad \textbf{(3.8)}$$

$$\boldsymbol{LB_j} = \boldsymbol{\Psi_j^{lower}}$$
(3.9)

Step 1: Random generate Male ant L_h as shown in equation (3.10) below:

$$[\boldsymbol{L_h}]_{C_i} = [\boldsymbol{l_1}, \dots \boldsymbol{l_j}, \dots \boldsymbol{l_N}]_{C_i} \quad \textbf{(3.10)}$$

Where,

$$\boldsymbol{l_j = LB_j + rand \times (UB_j - LB_j)} \quad \textbf{(3.11)}$$

$$\boldsymbol{rand} \sim \cup\,([\boldsymbol{0,1}]) \quad \textbf{(3.12)}$$

$$\boldsymbol{N} = \text{Number of dimension} \quad \textbf{(3.13)}$$

Step 2: Binary form conversion of N number of l_j for the male gene $[L_h]_{C_i}$ in every colony take place. The colonies' identity refers gene $[R_h]_{C_i}$ is form by complement of the binary code gene as shown as equation (3.14) below:

$$[\boldsymbol{R_h}]_{\boldsymbol{C_i}} = \overline{[\boldsymbol{L_h}]}_{\boldsymbol{C_i}} = \left[\boldsymbol{l_1}', \ldots\ \boldsymbol{l_j}', \ldots\ \boldsymbol{l_N}'\right]_{\boldsymbol{C_i}} \quad \textbf{(3.14)}$$

Then, every mated queen $[Q_d]_{C_i}$is formed as given in equation (3.6).

Step 3: Generate Offspring. This process representing mated queens generate offspring. The process is implemented by cloning the gene (L_h) binary bits from refer gene (R_h) start from the least significant bit (LSB) to the most significant bit (MSB) and vice-versa. Mathematical equation explanation can be represented as the following. If there is a 3 dimensions problem given, the gene $L_h$of colony 1 represented as $[L_h]_{C_1} = [l_1, l_2, l_3\ ]_{C_1}$ together with it's refer gene as $[R_h]_{C_1} = [l'_1, l'_2, l'_3\ ]_{C_1}$. As an example, by letting one of the dimension $l_1 = 10110010$ and the refer gene is the complement of it as $l'_1 = 01001101$. The offspring of the dimensions l_1 are generated as shown as the example below in Table 3.1 and Table 3.2:

Table 3.1: Offspring generated by LSB to MSB copying complement refer gene

l_1	$l_1{}'$
10110010	01001101
10110010	
1011001$\underline{1}$	
101100$\underline{01}$	-
10110$\underline{101}$	
⋮	
$\underline{01001101}$	

Table 3.2: Offspring generated by MSB to LSB copying complement refer gene

l_1	$l_1{}'$
10110010	01001101
10110010	
$\underline{0}$0110010	
$\underline{01}$110010	-
$\underline{010}$10010	
⋮	
$\underline{01001101}$	

Then, the same outcome of the offspring will be removed.

After 1st iteration, the refer gene (R_h) will remain, and the gene (L_h) will be updated. In this case, the matching will be $[L_h]_{C_1} = [l_1, l_2, l_3\ ]_{C_1}$ with the refer gene $[R_h]_{C_1} = [R_{h1}, R_{h2}, R_{h3}\ ]_{C_1}$. For example, let $l_1 = 11010011$ and refer gene still same as previous, $R_{h1} = 01001101$. The same outcome of the offspring will be removed. The dimensions l_1 of the offsprings are generated as shown below:

Table 3.3: Offspring generated by LSB to MSB copying other refer gene

l_1	$l_1{}'$
11010011	01001101
11010011 $1101001\underline{1}$ $110100\underline{01}$ $11010\underline{101}$ ⋮ $\underline{01001101}$	-

Table 3.4: Offspring generated by MSB to LSB copying other refer gene

l_1	$l_1{}'$
11010011	01001101
11010011 $\underline{0}1010011$ $\underline{01}010011$ $\underline{010}10011$ ⋮ $\underline{01001101}$	-

Step 4: The fittest offspring of every colony will be chosen as $\left[L_{h(fit)}\right]_{C_i}$ by assessing the fitness subjected to the objective function. Then the best fittest gene is selected among all the fittest offspring$\left[L_{h(fit)}\right]_{C_i}$ and stored as $L_{h(best)}$. The $L_{h(best)}$ is endorsed as the contemporary solution found. If the $L_{h(best)}$ recorded is repeated as many as Saturation Number, S times prefixed earlier. If the Saturation achieved, redirect to Step 6, otherwise redirect to Step 5.

Step 5: Nuptial Flight. In nuptial flight, males will fly away from their own colony to crossbreed with queens inhabited at other colony. This nuptial flight process is imitated by using the best offspring $\left[L_{h(fit)}\right]_{C_i}$ as the fittest survived male ant of every colony paring to the next colony refer gene $[R_h]_{C_{i+1}}$ as the inhabited queen. In the equation, the best offspring $\left[L_{h(fit)}\right]_{C_N}$ from the last colony will be paired with the refer gene $[R_h]_{C_1}$ from the first colony. This mechanism is represented in the equation (3.15):

In queen inhabited colony C_{i+1},

$$[\boldsymbol{L_h}]_{\boldsymbol{C_{i+1}}}[\boldsymbol{R_h}]_{\boldsymbol{C_{i+1}}} = \left[\boldsymbol{L_{h(fit)}}\right]_{\boldsymbol{C_i}}[\boldsymbol{R_h}]_{\boldsymbol{C_{i+1}}} \tag{3.15}$$

Where $i = 1,2, \dots N-1$

For the queen inhabited 1st colony C_1:

$$[\boldsymbol{L_h}]_{\boldsymbol{C_1}}[\boldsymbol{R_h}]_{\boldsymbol{C_1}} = \left[\boldsymbol{L_{h(fit)}}\right]_{\boldsymbol{C_N}}[\boldsymbol{R_h}]_{\boldsymbol{C_1}} \tag{3.16}$$

This process will go to Step 3 for generating new offspring. The process of Step 3 to Step 5 will be repeated until saturation S acheaved.

Step 6: The shrinking factor r is used to shrink the search boundary by positioned the $L_{h(best)}$ as the center of the boundary. The equations below show the shrinking calculation.

$$\textbf{Boundary size, } \boldsymbol{B_j = r \times \left(UB_j - LB_j\right)} \tag{3.17}$$

Then UB_j and LB_j are generated as shown below:

$$\boldsymbol{UB_j = \left[l_j\right]_{L_{h(best)}} + \frac{1}{2}B_j} \tag{3.18}$$

$$\boldsymbol{LB_j = \left[l_j\right]_{L_{h(best)}} - \frac{1}{2}B_j} \tag{3.19}$$

Step 7: New queens are produced for leading new colonies after certain times nuptial flights happened in all colony. This new queen creation process is performed by reproducing all the haploid gene $[L_h]_{C_i}$ and refer gene $[R_h]_{C_i}$ same as in Step 1 and continue with Step 2. In this new colonies generated in Step 1 and Step 2, one of the refer gene $[R_h]_{C_i}$ is replaced with current generation's best gene $L_{h(best)}$. This shows the fittest queen survived and brings forward the fittest gene to the next generation. The equation of fittest gene replacement process is shown as:

$$[\boldsymbol{R_h}]_{\boldsymbol{C_{random}}} = \boldsymbol{L_{h(best)}} \tag{3.20}$$

The evolution process could be considered converged when there is no consequential improvement in $f(L_{h(best)})$. The best gene $L_{h(best)}$ is taken as the final solution for the optimization problem if either of the 2 criteria listed below is valid or else the algorithm continues from Step 2 to 7 until convergences to minimum solution:

- If maximum number of iterations G exceeded.
- If SDAA saturated by satisfying these conditions shown below:

$$\left\|\max\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g}} - \max\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g-1}}\right\| \leq \boldsymbol{\varepsilon} \tag{3.21}$$

And

$$\left\|\min\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g}} - \min\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g-1}}\right\| \leq \boldsymbol{\varepsilon} \tag{3.22}$$

And

$$\left\|\max\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g}} - \min\left(\boldsymbol{f}(\boldsymbol{L_{h(best)}})\right)^{\boldsymbol{g}}\right\| \leq \boldsymbol{\varepsilon} \tag{3.23}$$

The pseudo code of SDAA is shown below and SDAA optimization flowchart is shown in Figure 3.2.

```
VARIABLES:
  dimension = number of dimension that problem defined.
  ub, lb, boundary= upper and lower bound, boundary defined by problem
  current boundary= boundary shrunken and used in every iteration in the program
  shrinking factor, r = between 0 and 1
  shrinking limit= minimum shrunken boundary
  input digit precision = input decimal size used for function evaluator
  max iteration = limit of iteration
  no. of colony = no. of solution take part in the program
  max inner loop = limit of inner loop
  max saturation count = to stop inner loop when saturated
 ε = convergence parameter to check the maximum convergence
 function_evaluator () = objective function

MAIN PROGRAM
START
 //low precision input for 1st iteration
Input parameters: input digit precision, dimension, number of colony, ub, lb, boundary, r, ε, maximum iteration, maximum inner loop, saturation count, shrinking limit.
  boundary = ub – lb
  current boundary = boundary
```

```
//generate random $[L_h]_{(C(i))}$ as initial solutions
$[L_h]_{(C(i))}$ = random × current boundary + lb
**Call** bit size calculation(**$L_h$**); //count bit size needed
**Call** floating_to_binary convertion(**$L_h$**);
Best current L = **min** function_evaluator (**$L_h$**)
$R_h$ = complement of (**$L_h$**);

While (iteration <= max iteration OR boundaries > shrinking limit OR ε satisfied)
    While (inner loop < max inner loop OR saturation count<max saturation count)
        Diploid form, $O_d$= $[L_h]_{(C(i))}$ $[R_h]_{(C(i))}$
        For colony= 1 to number of colony
            For variable= 1 to no. of dimension
                Generate offspring using copying process as describe in Step 3
            EndFor

            //test the problem function using **$L_{h(new)All}$**
            f( $L_{h(best)}$)= **min** function_evaluator (**$L_{h(new)All}$** );
             **Nuptial Flight** as describe in **Step 5**
        EndFor
        saturation count  // check saturation
    EndWhile
    Best current L= minimum result from all colonies $[L_{(h(best))}]_{(C(i))}$
    Check: iteration = max iteration? Boundary < shrinking limit? ε satisfied?
```

```
   If Check == yes for any of it
           final answer found as **Best current L**
        **break;**
   End If
    //before new iteration
    current range=r × current range;
    ub = Best current L + 0.5 × current range
    lb = Best current L - 0.5 × current range
    boundary =ub-lb
    **Call** bit size calculation($\mathbf{L_h}$);       //count bit size needed
    **Call** floating_to_binary convertion(**Best current L**);
    $[L_h]_{(C(i))}$ = random × current boundary + lb;   //generating new  $L_h$
     $[R_h]_{(C(i))}$ = complement of ($[L_h]_{(C(i))}$ ); // generate new queens
    rand = random integer between number of colony
     $[R_h]_{C(rand)}$= Best current L;  //bring forward fittest gene to next generation
  End While
STOP
```

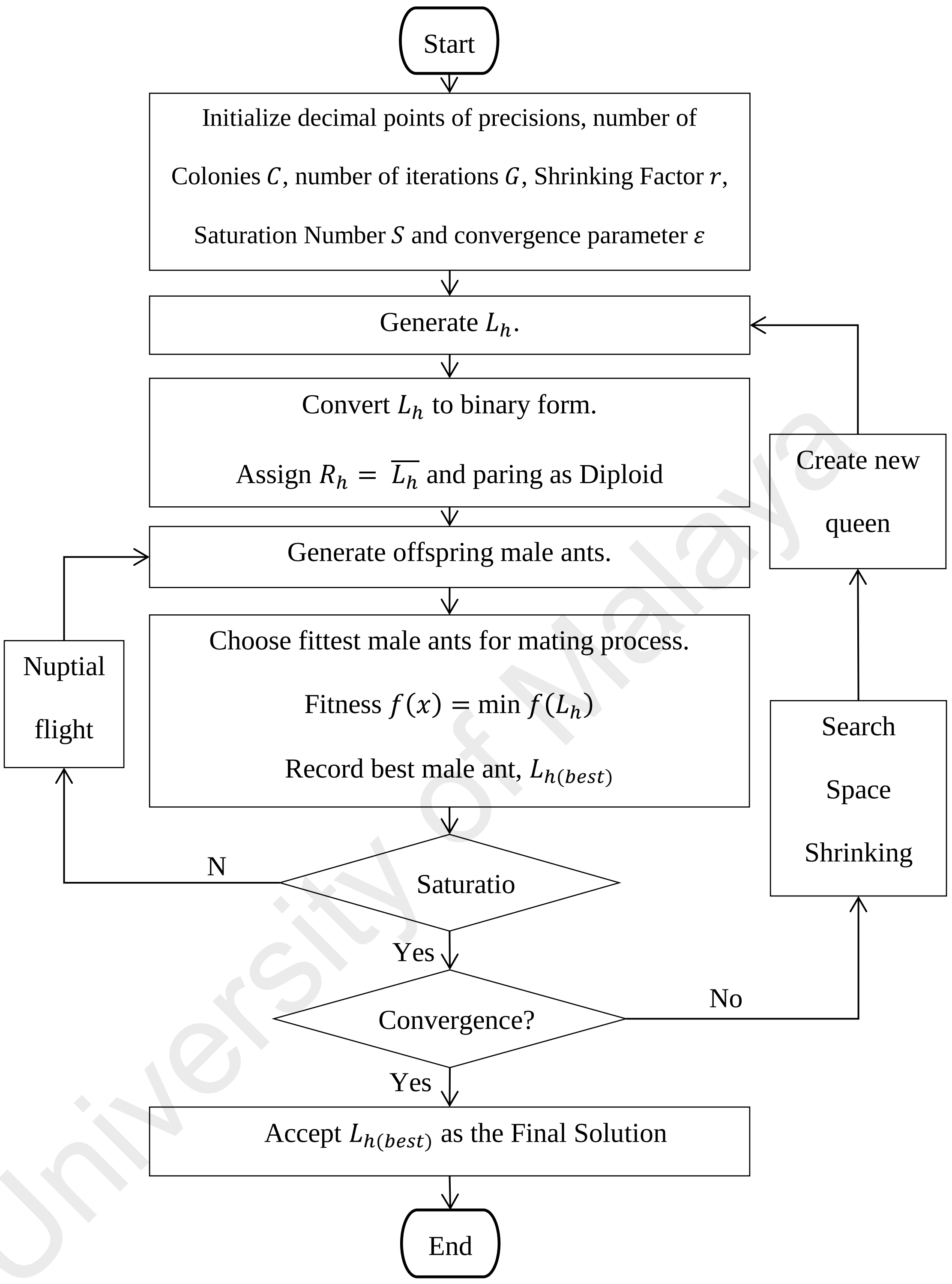
Start
Initialize decimal points of precisions, number of Colonies C, number of iterations G, Shrinking Factor r, Saturation Number S and convergence parameter ε
Generate L_h.
Convert L_h to binary form.
Assign $R_h = \overline{L_h}$ and paring as Diploid
Generate offspring male ants.
Choose fittest male ants for mating process.
Fitness $f(x) = \min f(L_h)$
Record best male ant, $L_{h(best)}$
Saturatio
N
Nuptial flight
Yes
Convergence?
No
Search Space Shrinking
Create new queen
Yes
Accept $L_{h(best)}$ as the Final Solution
End

Figure 3.2: Flowchart of SDAA

3.2 Modified Seed Disperser Ant Algorithm for Data Clustering

K-means clustering is a type of unsupervised learning, which able to solve or cluster the data given without defined categories or groups. K-means clustering is aiming to find and groups the data and the number of groups given. Based on the features of every data point provided K-means continually assign every data point to one of the groups. All the data points are clustered based on feature similarity. K-means clustering target to partition the data points into clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

The SDAA is modified name as Modified Seed Disperser Ant Algorithm (MSDAA) (Chang, Kanesan, Kulkarni, & Ramiah, 2017) to achieve improvement in terms of accuracy along with reducing function evaluations. This is implemented by modifying the Step 1 and cutting out Step 6 and Step 7 from SDAA. The control parameters such as number of colony C, decimal points of precisions for input variables, and saturation number S are set in the initialization stage. The upper boundary UB_j and lower boundary LB_j are generated same as original SDAA.

In Step 1, K-means search is used rather than using random creating initial solutions. The centroid found by using K-means search is used as the refer point to generate Male ant gene L_h. The search boundary is shrink by resized to 75% of the original boundary. The male ant gene L_h is produced randomly around the reference point within the search boundary.

Next, continue with the same steps from Step 2 to Step 5 as in original SDAA. $L_{h(best)}$ will be chosen as the final solution once Saturation in Step 5 is achieved. Step 6 and Step 7 of the original SDAA are cut out as it consumes large number of function evaluations. The modification implemented on SDAA resulted in impressive reductions of function evaluations for data clustering in comparison with original SDAA as well as others optimization algorithms. MSDAA is developed by adopting K-means search potential solution that speed up the overall search process and remove the excess steps of new queen spawn iteration from original SDAA. This enables MSDAA to obtain better result in shorter period of time in conjunction to SDAA as well as other algorithms.

In clustering, for a set of data, D with M data object is clustered to K sets of clusters:

$$\boldsymbol{D} = [\boldsymbol{Y_1}, \boldsymbol{Y_2}, \dots, \boldsymbol{Y_M}] \tag{3.24}$$

Where

$$\boldsymbol{Y_i} \in \boldsymbol{\Re^D} \tag{3.25}$$

$$\textbf{Clusters, } \boldsymbol{S} = [\boldsymbol{X_1}, \boldsymbol{X_2}, \dots, \boldsymbol{X_K}] \tag{3.26}$$

Each data in set D will be allocated in one of the K clusters in such way that it will minimize the objective function. The objective function, inter-cluster variance is defined as the sum of squared Euclidean distance between each object Y_I and the center of the cluster X_J which it belongs (Krishnasamy et al., 2014). This objective function is given by equation (3.27):

$$\boldsymbol{F}(\boldsymbol{X}, \boldsymbol{Y}) = \sum_{\boldsymbol{I}=\boldsymbol{1}}^{\boldsymbol{M}} \boldsymbol{Min} \left\| \boldsymbol{Y_I} - \boldsymbol{X_J} \right\|^{\boldsymbol{2}} \tag{3.27}$$

Where $J = 1, 2, \dots, K$

Also,

$$\boldsymbol{X_J} \neq \emptyset \tag{3.28}$$

Where $\forall J\{1, 2, \dots, K\}$

$$\boldsymbol{X_I} \cap \boldsymbol{X_J} \neq \emptyset \tag{3.29}$$

Where $\forall I \neq J$ and $\forall I, J\{1, 2, \dots, K\}$

$$\bigcup_{\boldsymbol{J}=\boldsymbol{1}}^{\boldsymbol{K}} \boldsymbol{X_J} = \boldsymbol{D} \tag{3.30}$$

For data clustering problems, the objective function $f(x) = F(X, Y)$ is represented by the sum of squared Euclidean distance between each object Y_I and the center of the cluster X_J as shown in equation (3.31).

$$\textbf{Minimize } \boldsymbol{f}(\boldsymbol{L_h}) = \boldsymbol{f}(\boldsymbol{l_1}, \dots\ \boldsymbol{l_j}, \dots\ \boldsymbol{l_N}) \tag{3.31}$$

Subject to

$$\boldsymbol{\Psi_j^{lower}} \leq \boldsymbol{l_j} \leq \boldsymbol{\Psi_j^{upper}} \tag{3.32}$$

Where $j = 1, \dots N$

In MSDAA, the Male ant gene $L_h = [l_1, \dots l_j, \dots l_N]$ is representing the center of the cluster, where dimension $X_J = \left[x_1, \dots x_j, \dots x_N\right]$ is represented by l_j.

$$\text{Where } \boldsymbol{L_h} = \left[\boldsymbol{x_1}, \dots \boldsymbol{x_j}, \dots \boldsymbol{x_N}\right] = [\boldsymbol{l_1}, \dots \boldsymbol{l_j}, \dots \boldsymbol{l_N}] \tag{3.33}$$

Thus, the objective is shown as:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{F}(\ \boldsymbol{L_h}, \boldsymbol{Y}) \tag{3.34}$$

The K-means Clustering algorithm work in few steps. First, k-means clustering the data into k groups where k is predefined. Second, random k points are selected as cluster centres. Third, objects that closest to the cluster centres are assigned to the cluster according to the Euclidean distance function. Forth, calculate the centroid or mean of all objects in each cluster. Then, repeat the second step to forth step until the same points are assigned to each cluster in consecutive rounds. Maximum iteration as a limit to prevent infinite looping.

The K-means pseudo code is shown below:

```
Input:    $D = \{ D_1, D_2, D_3, \dots, D_M\}$
        $K$ = number of clusters
        $Max\_Iter$ = limit of iteration
Output: $C = \{ c_1, c_2, c_3, \dots, c_M\}$ (set of cluster centroids)
          $L = \{l(e) \mid e = 1, 2, \dots, M\}$ (set of cluster labels of $D$)

foreach $c_i \in C$ do
        $c_i \leftarrow e_j \in D$ (e.g. random selection)
end

foreach $e_i \in E$ do
        $l(e_i) \leftarrow argminDistance(e_i, c_j)\ j \in \{1, \dots, K\}$
end
$changed \leftarrow false$;
 $iter \leftarrow 0$;
```

```
repeat
    foreach $c_i \in C$ do
        $UpdateCluster(\, c_i);$
    end
    foreach $e_i \in E$ do
        $minDist \leftarrow argminDistance(e_i, c_j)\ j \in \{1, \dots, K\}$ ;
        If $minDist \neq\ l(e_i)$
            $l(e_i) \leftarrow minDist;$
            $changed \leftarrow true;$
        else
            $changed \leftarrow false;$
        end
    end
    $iter++;$
until $changed = true$ and $iter \leq Max\_Iter$
```

The output centroids of K-means, $C = \{c_1, c_2, c_3, \dots, c_M\}$ will bring into SDAA for optimization. Output centroids of K-means C will use as initial generated centroids for SDAA, $L_h = C$. Pseudo code of MSDAA is shown below and the flowchart of MSDAA is shown in Figure 3.3.

```
VARIABLES:
  dimension = number of dimension that problem defined.
  ub, lb, boundary= upper and lower bound, boundary defined by problem
  current boundary= boundary shrunken and used in every iteration in the
program
  input digit precision = input decimal size used for function evaluator
  no. of colony = no. of solution take part in the program
  max inner loop = limit of inner loop
  max saturation count = to stop inner loop when saturated
 $\varepsilon$ = convergence parameter to check the maximum convergence
function_evaluator () = objective function

MAIN PROGRAM
START
  //low precision input for 1st iteration
  Input parameters: input digit precision, dimension, number of colony, ub, lb,
boundary, r, $\varepsilon$, maximum inner loop, saturation count
  //Set upper and lower boundaries
  boundary = ub – lb
  current boundary = boundary
```

```
//generate $L_h$ using K-means as initial solutions
$L_h$= output centroids of K-means, $C = \{c_1, c_2, c_3, \ldots, c_M\}$
Call bit size calculation($L_h$); //count bit size needed
Call floating_to_binary convertion($L_h$);
$R_h$ = complement of ($L_h$);

While (inner loop < max inner loop OR saturation count<max saturation count)
    For colony= 1 to number of colony
        For variable= 1 to no. of dimension
            Generate offspring using copying process as describe in Step 3
        EndFor
        //test the problem function using $L_{h(new)All}$
        f( $L_{h(best)}$)= min function_evaluator ($L_{h(new)All}$ );
        Nuptial Flight as describe in Step 5
    EndFor
    saturation count  // check saturation
EndWhile
Best current L= minimum result from all colonies $[L_{(h(best))}]_{(C(i))}$
final answer found as Best current L
STOP
```

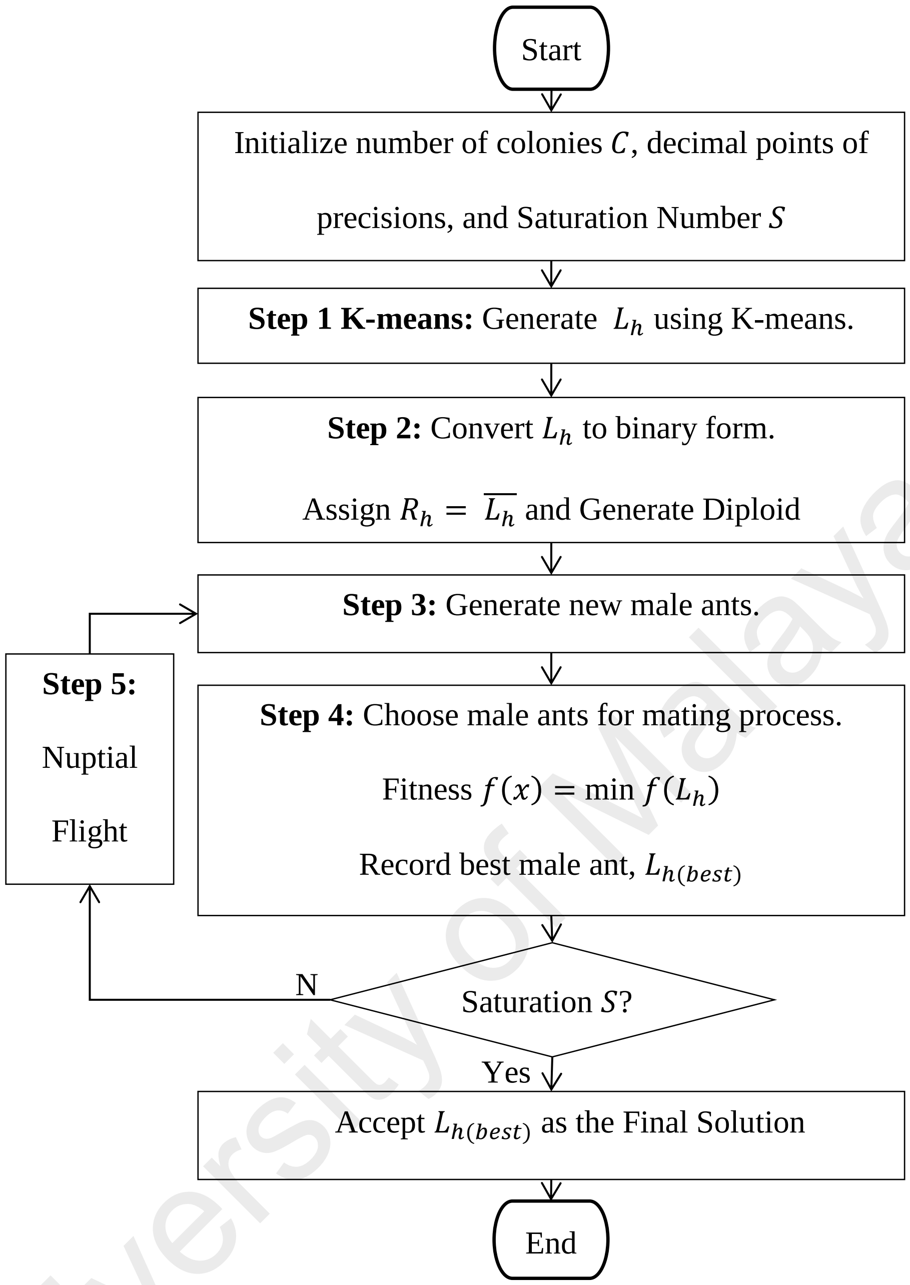
Start
Initialize number of colonies C, decimal points of precisions, and Saturation Number S
Step 1 K-means: Generate L_h using K-means.
Step 2: Convert L_h to binary form.
Assign $R_h = \overline{L_h}$ and Generate Diploid
Step 3: Generate new male ants.
Step 4: Choose male ants for mating process.
Fitness $f(x) = \min f(L_h)$
Record best male ant, $L_{h(best)}$
Step 5: Nuptial Flight
Saturation S?
N
Yes
Accept $L_{h(best)}$ as the Final Solution
End

Figure 3.3: Flowchart of MSDAA

Comparing flowchart of SDAA on Figure 3.2 and flowchart of MSDAA on Figure 3.3, SDAA contain 2 different loops where MSDAA only have 1 loop. K-means' result provide a good starting point and continue with SDAA search. This improves the efficiency and accuracy of the solution. Initial convergence is done by K-means. With the converged solution, SDAA explore to get better solution until saturation.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Classical and Composite Benchmark Problem Solving

30 dimensional benchmark problems of Ackley function (Adorio & Diliman, 2005), Rastrigin function (Molga & Smutnicki, 2005), Rosenbrock function (Shang & Qiu, 2006) and Sphere function (Molga & Smutnicki, 2005) are used for SDAA validation. Rosenbrock function has the minimum in a deep and narrow parabolic valley with a flat bottom. A large number of iterations needed for the gradient based methods to achieve the global minimum (Shang & Qiu, 2006). Sphere function is unimodal and strongly convex function. Ackley function is highly multimodal with unique global minimum (Kulkarni & Tai, 2009). The Rastrigin function has a lot of local minima. It is highly multimodal, but the minima locations are regularly distributed (Xu et al., 2013). These functions have its own difficulty to be optimized and are commonly used as benchmarks in literature to evaluate the performance of optimization techniques. 30 times of simulations were executed using MATLAB version R2012a in computer with Intel i7-4770 (3.40GHz) processor and 12GB RAM run on Windows 7 operation system.

The results of PSO, IPSO, EPSO, GPSO, CPSO, CS and SDAA are shown in Table 4.1. By comparing the results between all this optimization techniques, the results show SDAA is obtained better result. The average function evaluations (FEs) of SDAA achieving these results are recorded in Table 4.1. Most of the result shows that close to theoretical global minimum is obtained by SDAA.

From Table 4.1, SDAA use more FEs to solve Rastrigin function and Rosenbrock function. This is because SDAA need more exploratory search for exact minimum point out of a lot of local minima for Rastrigin function and also the process search through the minimal valley on the Rosenbrock function. The higher the FEs needed, the longer the time used for aching the solution. The result of others algorithm in Table 4.1 is referred from articles, this table is mainly to compare the best, mean, and the standard deviation (Std.) of the result found, FEs and time is recorded for future research comparison.

Table 4.1: Simulation Result of 30 dimensional test problems

		Sphere	**Rosenbrock**	**Rastrigin**	**Ackley**
PSO	**Best**	1.22E+03	1.79E+03	9.38E+01	9.48E+00
	Mean	3.70E+03	1.17E+04	1.48E+02	1.27E+01
	Std.	1.53E+03	7.31E+03	2.63E+01	1.45E+00
IPSO	**Best**	8.71E+02	7.16E+02	6.06E+01	7.48E+00
	Mean	2.25E+03	5.47E+03	1.19E+02	1.00E+01
	Std.	8.05E+02	5.08E+03	2.41E+01	1.16E+00
EPSO	**Best**	5.40E+03	1.62E+04	1.36E+02	1.30E+01
	Mean	1.14E+04	9.36E+04	2.35E+02	1.63E+01
	Std.	3.85E+03	7.21E+04	3.69E+01	1.14E+00
GPSO	**Best**	1.47E-10	5.36E-07	4.47E-07	5.59E-05
	Mean	4.51E-06	3.26E-05	2.96E-05	4.23E-04
	Std.	6.10E-06	5.57E-05	4.28E-05	2.83E-04
CPSO	**Best**	1.43E-81	1.34E-05	0.00E+00	6.33E-07
	Mean	3.04E-12	4.81E-02	2.52E-03	1.38E-04
	Std.	N/A	N/A	N/A	N/A
CS	**Best**	4.29E-15	N/A	1.77E-15	1.65E-07
	Mean	6.04E-13	N/A	4.72E-09	1.10E-06
	Std.	N/A	N/A	N/A	N/A
SDAA	**Best**	0.00E+00	1.74E-07	0.00E+00	8.88E-16
	Mean	0.00E+00	4.12E-05	0.00E+00	8.88E-16
	Std.	0.00E+00	7.70E-05	0.00E+00	0.00E+00
	FEs	11625	54934	22270	16375
	Time (s)	13.29	160.08	19.67	16.64

The Composite benchmark functions from Congress on Evolutionary Computation (CEC) 2005 special session (Liang, Suganthan, & Deb, 2005) are shown in Appendix A also solved using SDAA. These Composite benchmarks are 10 dimensional benchmarks. The Composite benchmark functions are complicated optimization problems. These benchmarks are challenging to be solve as they are expanded, shifted, rotated, and merged with multiple variation of the classical functions. Furthermore these benchmark functions having a lot local optima which provide higher complexity.

The results of Composite benchmark functions are recorded in Table 4.2. There are several benchmarks with better result are shown by comparing the results of SDAA with GWO, PSO, GSA, DE and CMA-ES. As specially CF1, CF2 and CF5 which SDAA able to obtain lowest mean result and lowest standard deviation (Std.) among all this algorithms. As No Free Lunch Theorem stated, no algorithm able to work perfectly on every problem. In Table 4.2, none of an algorithm successfully solved all the Composite benchmark functions. SDAA perform well on 3 out of 6 Composite benchmark functions. Each Composite benchmark function is formed by combining different kinds of conditions. The combination of different conditions problem becomes a big challenge for every type of algorithms to solve.

Table 4.2: Result of composite benchmark functions

		CF1	CF2	CF3	CF4	CF5	CF6
SDAA	**Mean**	2.88E-17	16.4943	179.3623	335.8192	6.4585	457.4799
	Std.	1.09E-17	6.4343	40.6356	77.7292	2.5709	43.3633
GWO	**Mean**	43.8354	91.8008	61.4377	123.1235	102.1429	43.1426
	Std.	69.8614	95.5518	68.6881	163.9937	81.2553	84.4857
PSO	**Mean**	100	155.91	155.91	314.3	83.45	861.42
	Std.	81.65	13.176	13.176	20.066	101.11	125.81
GSA	**Mean**	6.63E-17	200.6202	180	170	200	142.0906
	Std.	2.78E-17	67.7208	91.8936	82.3272	47.1404	88.8714
DE	**Mean**	6.75E-02	28.759	144.41	324.86	10.789	490.94
	Std.	1.11E-01	8.6277	19.401	14.784	2.604	39.461
CMA-ES	**Mean**	100	161.99	214.06	616.4	358.3	900.26
	Std.	188.56	151	74.181	671.92	168.26	8.32E-02

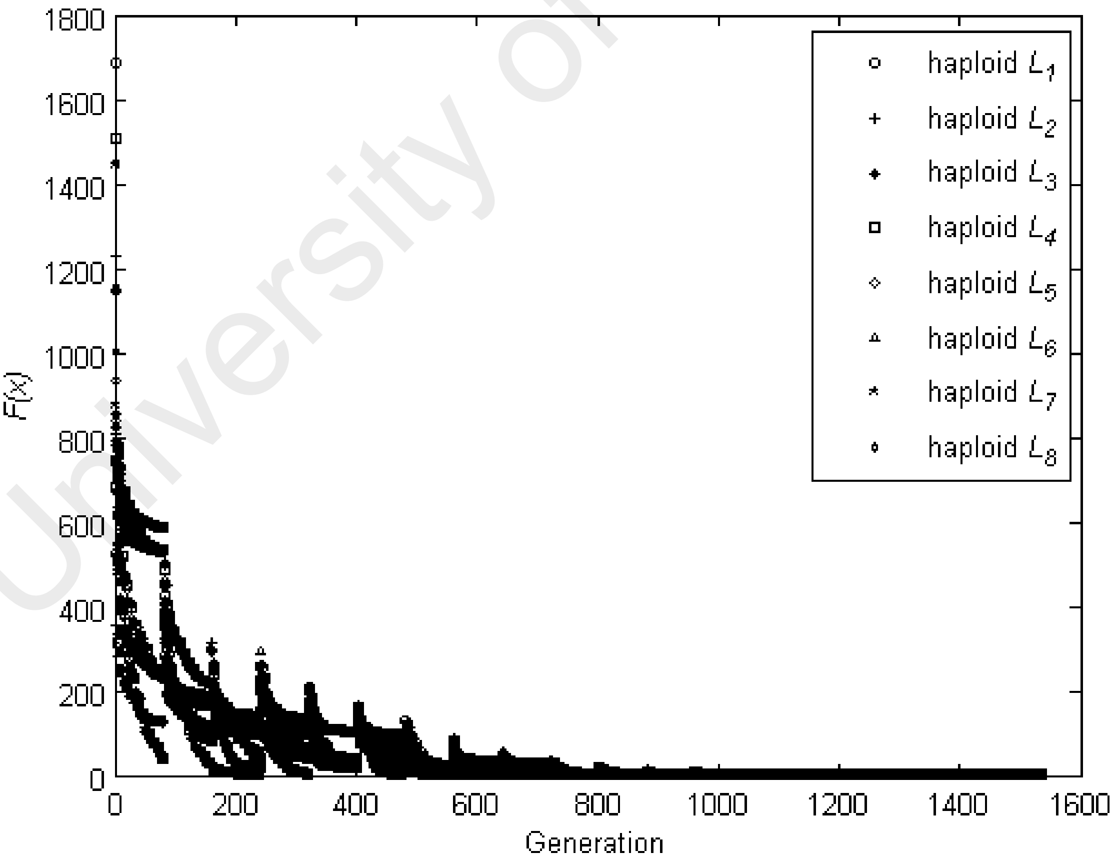

Figure 4.1: Plot for Composite benchmark functions CF1

Figure 4.1 shows the plot for Composite benchmark functions CF1 using SDAA. From the shape of the plot, every haploid are converging to its saturation point. Exploration start again the search, then continue on convergence to its saturation point. The exploration search and exploitation search repeated until all the saturation point come to the same. There are a lot of peak on every start of exploration search, the enlarged plot is shown on Figure 4.2 or better vision.

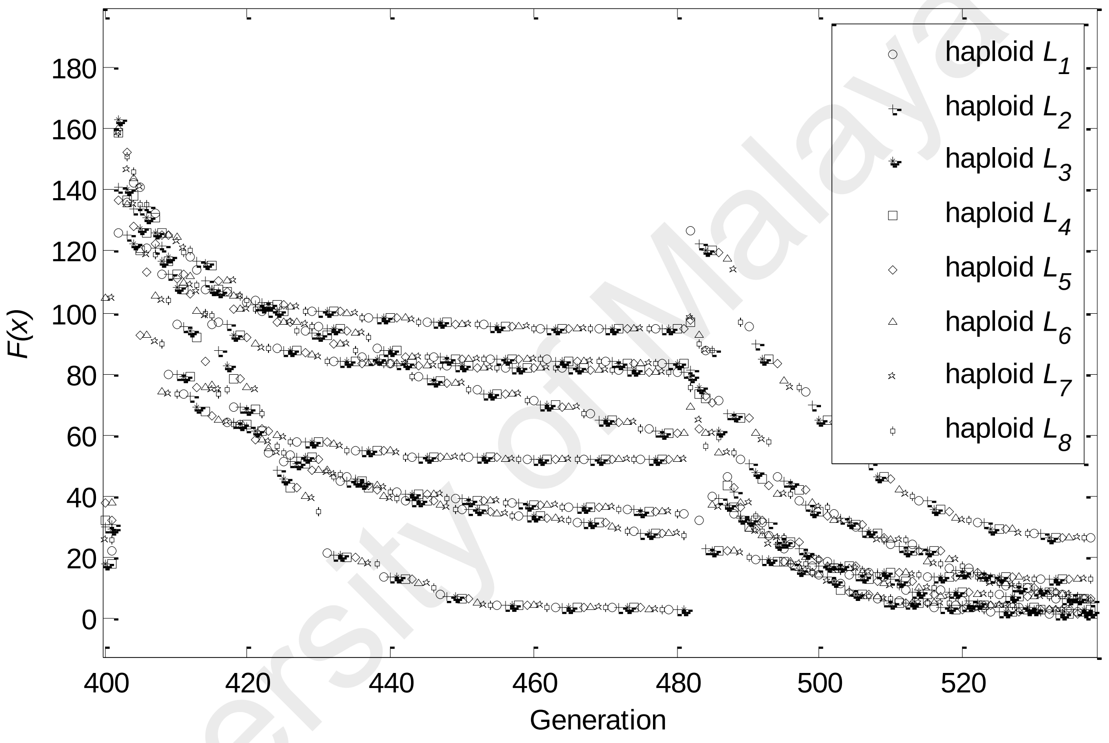

Figure 4.2: Enlarged plot of Composite benchmark functions CF1

Figure 4.2 shows the enlarged version of Figure 4.1 between 400th to 540th generations. Nuptial flight process can be observed in Figure 4.2 where the fitness haploid of male ant represented by L_h change by generations. Subsequently, the peak in the plot is due to migration of new fit queen and random colony generated near new fit queen in between the shrunken boundary. The random generated colonies' queens and male ant shows minor reduction in fitness, however it exploring and converged to better solutions.

Convergence towards the best solution in SDAA is done using a shrinking process. This shrinking process is control by a parameter namely shrinking factor. The search will shrinks towards the fittest haploid value in the given fixed boundary. SDAA expend local search to decrease the duty of localize search accountable by shrinking factor. All SDAA simulation shrinking factor is set to 0.75 and the colony size is set to 8 on every test benchmarks in this research. To expressing altruistic behaviour, other haploids referring the best haploid found as refer gene to enhance their corresponding fitness. In other hand, the best haploid also able to enhance its fitness by referring less fit haploid as an exploration search. This lead SDAA achieved the global optimum by all haploids locate global fitness via robust local search on altruistic haploid.

4.2 Constrained Engineering Problem Solving

The single-objective test problem (Floudas & Pardalos, 1990) used in many research (Ben Hadj-Alouane & Bean, 1997; Ben Hamida & Schoenauer, 2002; Coello & Cortés, 2004; Koziel & Michalewicz, 1999; Michalewicz & Attia, 1994; Yıldız, 2009; Yoo & Hajela, 1999) as test benchmark has 13 variables and 9 inequality constraints. Tension spring design problem was described by Belegundu (Belegundu & Arora, 1985) as minimizing the weight of a tension spring subject to constraints on minimum deflection, shear stress, surge frequency and outer diameter. The design variables are the mean coil diameter x_1, wire diameter x_2 and number of active coils x_3. Pressure vessel design problem is taken from Kannan and Kramer design problem (Kannan & Kramer, 1994). This design problem is based on a cylindrical vessel capped at both end with hemispherical heads. The objective is to minimize the total cost, include the cost of material, forming and welding. Four design variables include thickness of the shell x_1, thickness of the head x_2, inner radius x_3, and length of cylindrical section of the vessel (not include heads) x_4. The thickness of the shell and head, x_1 and x_2 are integer multiply with 0.0625 inch, which are the available thickness of rolled steel plates. Inner radius x_3, and length of cylindrical section of the vessel x_4 is continuous variables. From the minimum result found, x_1 and x_2 is rounded up to the closest valid thickness (integer multiply with 0.0625 inch). SDAA continues the optimization process by using x_1 and x_2 found. Available mathematical analysis of this problem proves that 6,059.7143 is the global minimum (X.-S. Yang, Huyck, Karamanoglu, & Khan, 2013). Welded beam design problem is firstly proposed by Coello (Coello Coello, 2000). A welded beam is designed for minimum cost subject to constraints on shear stress τ, bending stress in the beam σ, buckling load on the bar P_C, end deflection of the beam δ, and side constraints with the four design variables x_1, x_2, x_3, and x_4 (Rao & Rao, 2009).

SDAA shows the best result with standard deviation, Std. = 0 and lowest FEs in comparison with other algorithms in Table 4.3. For the result of tension spring design problem in Table 4.4, SDAA has the best result equal to RHPSO's best result which is the minimum found so far. RHPSO and DELC have slightly better mean result compared to SDAA in tension spring design problem. However, SDAA used lowest FEs of 17125 to solve tension spring design problem in comparison with all algorithms shown in Table 4.4. Table 4.5 show the result comparison for pressure vessel design problem. SDAA able to achieve best result and best mean result with lowest Std. and FEs. For the result of welded beam design problem in Table 4.6, SDAA shows the best and mean result and lowest FEs compared to all others algorithms.

The sample convergence histories for two problems namely Tension Spring Design Problem and Welded Beam Design Problem are presented in Figure 4.3 and Figure 4.5 respectively. The position of L_h of each colony in every nuptial flight in the convergence plot shows the fitness L_h that mated with the current colony queen. The spike or peak of the convergences plots shows the new queen generation which regenerate all colonies with shrunken search space. The spike can be observed clearly in magnified convergence plots in Figure 4.4 and Figure 4.6.This perturbation helps the optimization to escape local optima. Despite the perturbation, the solution converged to the minimum with rapid improvement as shown in Figure 4.3 convergence of the Tension Spring Design Problem. This is similar in Figure 4.5 that shows the convergence of Welded Beam Design Problem. Again, the fitness/solution of male ants in every new colony was improved compared to previous colony as emergence of young queen/ colony helps to improve the fitness of male ants. In SDAA, both nuptial flight and young queen production helps the optimization process in terms of exploitation and exploration respectively.

Table 4.3: Result of Single Objective Test Problem

Methods	Best	Mean	Worst	Std.	FEs
SDAA	**-15.000000**	**-15.000000**	**-15.000000**	**0**	**2812**
RHPSO (Xu et al., 2013)	**-15.000000**	-14.921875	-13.828125	0.297314	100000
PSO (Yıldız, 2009)	**-15.000000**	-14.876000	-14.681900	0.113000	100000
Hybrid GA (Coello & Cortés, 2004)	-14.784100	-14.526600	-13.841700	0.233500	150000
Modified GA (Yoo & Hajela, 1999)	-5.273500	-3.743500	-2.425500	0.969600	150000
ASCHEA (Ben Hamida & Schoenauer, 2002)	**-15.000000**	-14.840000	N/A	N/A	1500000
Homomorphous Mappings (Koziel & Michalewicz, 1999)	-14.786400	-14.708200	-14.615400	N/A	1000000
GA (Ben Hadj-Alouane & Bean, 1997)	-5.1655900	-3.6400400	-2.7251800	0.6062400	N/A
Genocop II (Michalewicz & Attia, 1994)	-7.3433400	-5.0713600	-3.5953600	0.7724700	N/A

The result in Table 4.3 shows that only SDAA able to achieve 0 standard deviation. This mean SDAA able to solve Single Objective Test Problem every time without error. Result from other algorithms show that they have chances to trap in local minima that not always achieved the best result or theoretical result. The Result also shows SDAA have the advantage to achieve the best result in low FEs.

Table 4.4: Result of Tension Spring Design Problem

Methods	Best	Mean	Worst	Std.	FEs
SDAA	**0.01266523**	0.01266975	0.01267941	5.66E-06	**17125**
RHPSO (Xu et al., 2013)	**0.01266523**	**0.01266523**	**0.01266524**	**1.54E-09**	30000
PSO (Yıldız, 2009)	0.01266527	0.01267300	0.01270800	6.24E-06	30000
Hybrid GA (Coello & Cortés, 2004)	0.01268100	0.01274200	0.01297300	5.90E-05	80000
Self-Adaptive Penalties GA (Coello Coello, 2000)	0.01270480	0.01276900	0.01282200	3.94E-05	900000
DE (Lampinen, 2002)	0.0126702	0.012703	0.012790	2.7E−05	204800
DELC (Wang & Li, 2010)	**0.01266523**	0.01266527	0.01266558	1.3E−07	20000
CPSO (He & Wang, 2007a)	0.0126747	0.0127300	0.0129240	5.20E−04	240000
(μ + λ)-ES (Mezura-Montes & Coello, 2005)	0.012689	0.013165	N/A	3.9E-04	30000

Tension Spring Design Problem is giving some challenge to SDAA. Although the mean result of SDAA not the best, SDAA still able to found the best result from 30 times of test done. DELCE also able to achieve the best result and perform slightly better than SDAA. RHPSO have the lowest standard deviation in Table 4.4 and the best result slightly higher compare SDAA and DELCE. SDAA and DELCE perform well in solving Tension Spring Problem that using a low number of FEs. The best result is important because in real world engineering problem, a slightly better result may affect the cost and quality of the production.

Table 4.5: Result of Pressure Vessel Design Problem

Methods	Best	Mean	Worst	Std.	FEs
SDAA	**6059.7143**	**6059.7143**	**6059.7143**	**0**	**28140**
RHPSO (Xu et al., 2013)	**6059.7143**	6059.7145	6059.7183	0.0007	30000
PSO (Yıldız, 2009)	6059.7144	6097.4460	6156.5700	35.7810	30000
Hybrid GA (Coello & Cortés, 2004)	6061.1229	6734.0848	6738.0602	457.9959	150000
Constraint-Handling GA (Coello Coello & Mezura Montes, 2002)	6059.9463	6177.2533	6469.3220	130.9297	80000
Self-Adaptive Penalties GA (Coello Coello, 2000)	6288.7445	6293.8432	6308.1497	7.4133	900000
CPSO (He & Wang, 2007a)	6061.0777	6147.1332	6363.8041	86.45	240000
HPSO (He & Wang, 2007b)	**6059.7143**	6099.9323	6288.6770	86.20	81000
CDE (Huang, Wang, & He, 2007)	6059.7340	6085.2303	6371.0455	43.0130	204800

In Table 4.5, SDAA again show its potential that solving Pressure Vessel Design Problem with best result and 0 standard deviation. No other algorithms able to have persistency to get the best result as SDAA in solving this problem. Although RHPSO and HPSO able to achieve the best result, the FEs used in solving this problem is higher compare SDAA. FEs is taking into account to reduce the time and computation power used for solving the problem.

Table 4.6: Result of Welded Beam Design Problem

Methods	Best	Mean	Worst	Std.	FEs
SDAA	**1.723703**	**1.724654**	1.728197	0.00453	**18197**
Self-Adaptive Penalties GA (Coello Coello, 2000)	1.748309	1.771973	1.785835	0.0112	900000
Constraint-Handling GA (Coello Coello & Mezura Montes, 2002)	1.728226	1.792654	1.993408	0.0747	80000
CAEP (Coello Coello & Becerra, 2004)	1.724852	1.971809	3.179709	0.443	50020
CPSO (He & Wang, 2007a)	1.728024	1.748831	1.782143	0.0129	240000
HPSO (He & Wang, 2007b)	1.724852	1.749040	1.814295	0.0401	81000
PSO-DE (H. Liu, Cai, & Wang, 2010)	1.724852	1.724852	**1.724852**	**6.7E−16**	66600
NM-PSO (Zahara & Kao, 2009)	1.724717	1.726373	1.733393	0.00350	80000
DE (Lampinen, 2002)	1.733461	1.768158	1.824105	0.0221	204800
CDE (Huang et al., 2007)	1.73346	1.76815	N/A	N/A	240000
(μ + λ)-ES (Mezura-Montes & Coello, 2005)	1.724852	1.777692	N/A	0.088	30000
ABC (Akay & Karaboga, 2012)	1.724852	1.741913	N/A	0.031	30000

SDAA perform well in solving Welded Beam Design Problem as shown in Table 4.6. The FEs used is for more lower compare other algorithms. Although SDAA might get the worst solution that far away from the best solution. But the mean result is close to the best result shows that the chances to get worst result is low. This is the reason mean result is important on solving any optimization problems.

Table 4.7 Parameters used for SDAA for constrained engineering problem solving

C	S	G	r	ε
8	8	300	0.90	1E-05

Table 4.7 record the parameters used in SDAA to solve all the constrained engineering problem. There is 5 parameter use to control SDAA behaviours. This is considered too many parameters to control and hard to achieve optimum conditions that able to solve all kinds of problem. Enhancement needed to reduce the parameter used and improve the performance.

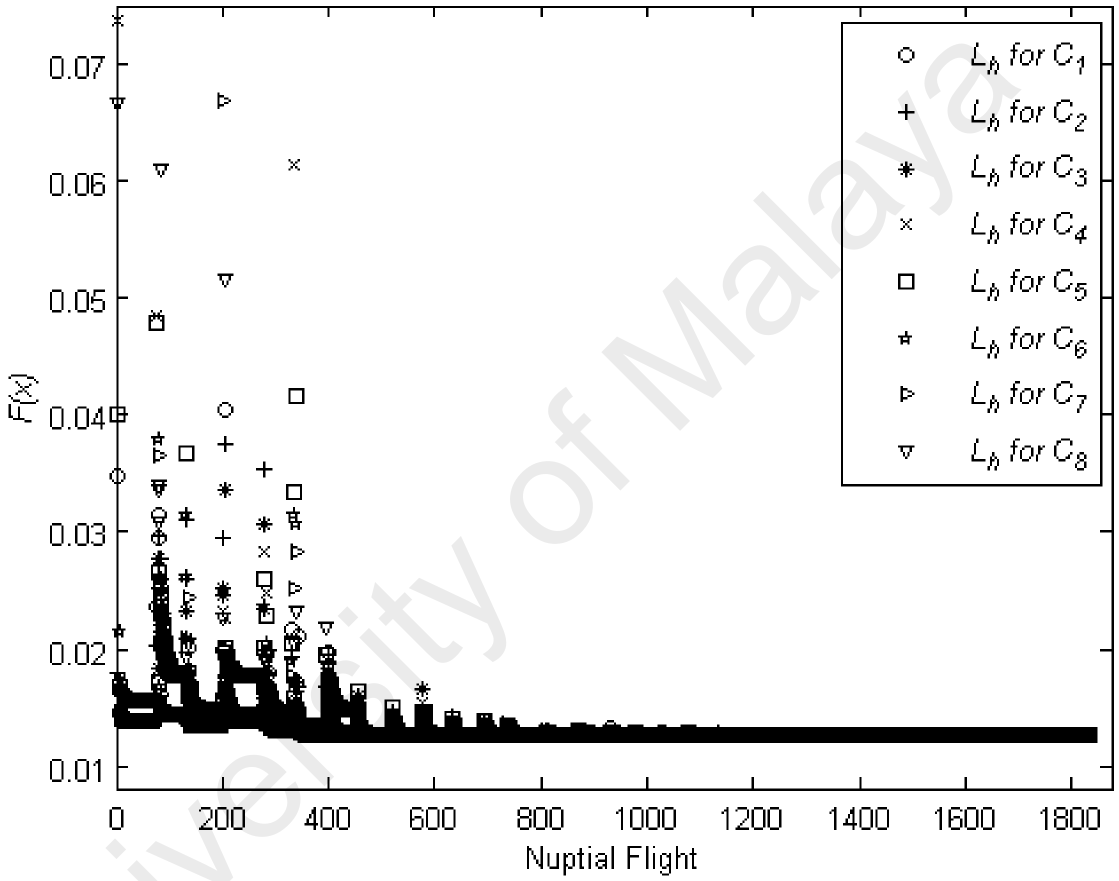

Figure 4.3: Convergence of Tension Spring Design Problem

Figure 4.3 shows the convergence of SDAA on solving Tension Spring Problem. The plot shows the peak on every exploration search far away from the minimum. This is the reason and behaviour of SDAA which cause the chances trap on local minima. Although SDAA not always getting the best result in solving Tension Spring Problem, SDAA still manage to get close to best result as shown in Table 4.4. Form Figure 4.3, the convergence of SDAA shown is very robust in the exploitation search.

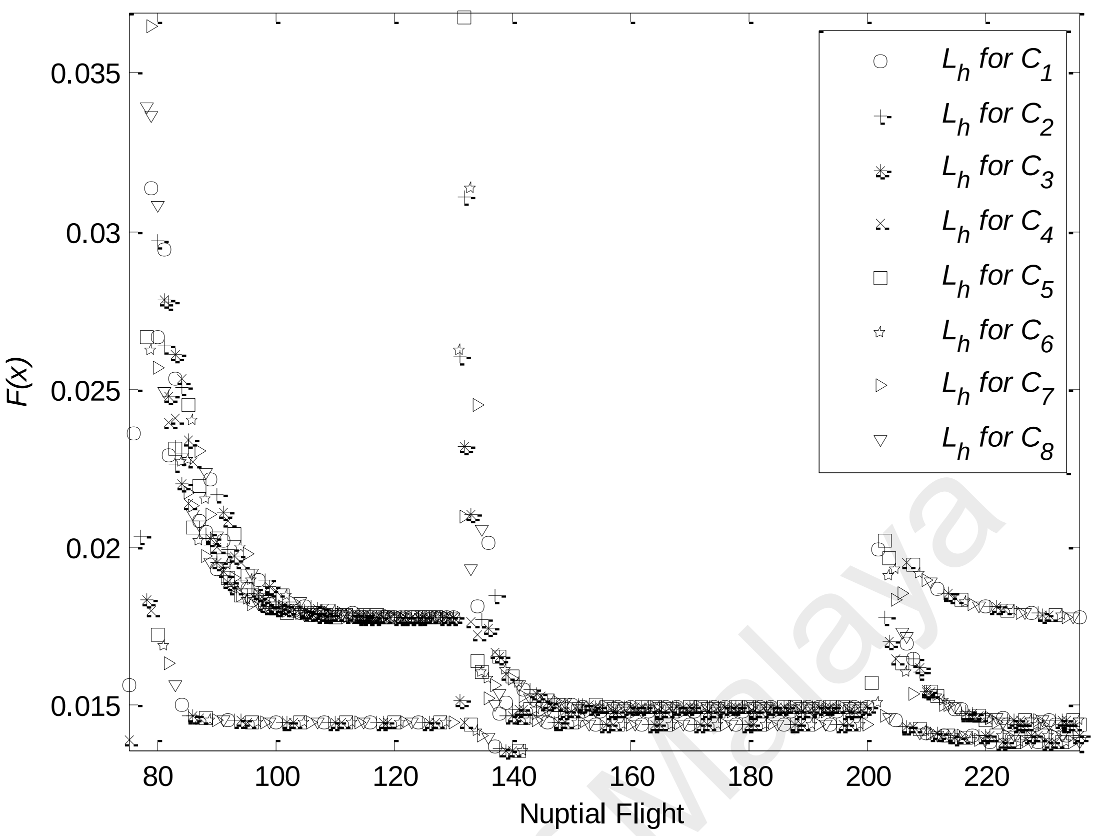

Figure 4.4: Magnified Convergence of Tension Spring Design Problem between nuptial flights 75 to 240

In Figure 4.4, the plot shows the enlarged or magnified plot of Figure 4.3. From this plot, the different haploid started different point at the start of every exploration. Almost all haploid converged and saturated to the same saturation point and next exploration search will start again. This behaviour of SDAA shown the exploration and exploitation search happened.

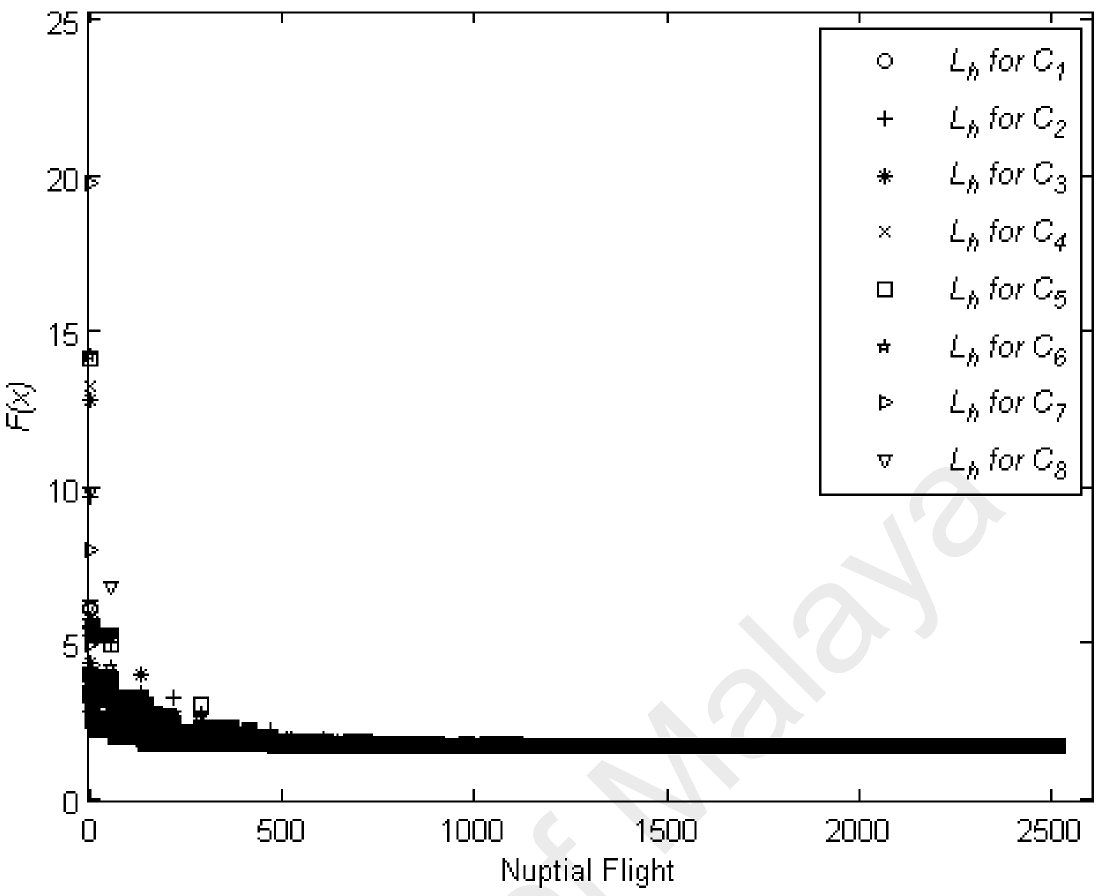

Figure 4.5: Convergence of Welded Beam Design Problem

Figure 4.5 shows the plot of SDAA solving Welded Beam Design Problem. The Plot shows that the convergence is fast and almost saturation just after 500 Nuptial Flight. This shows that SDAA easy to get near best solution for solving Welded Beam Design Problem. In another point of view, this is also the reason that SDAA might easily trap in local minima if the number of haploid is not enough to randomly explore the solution.

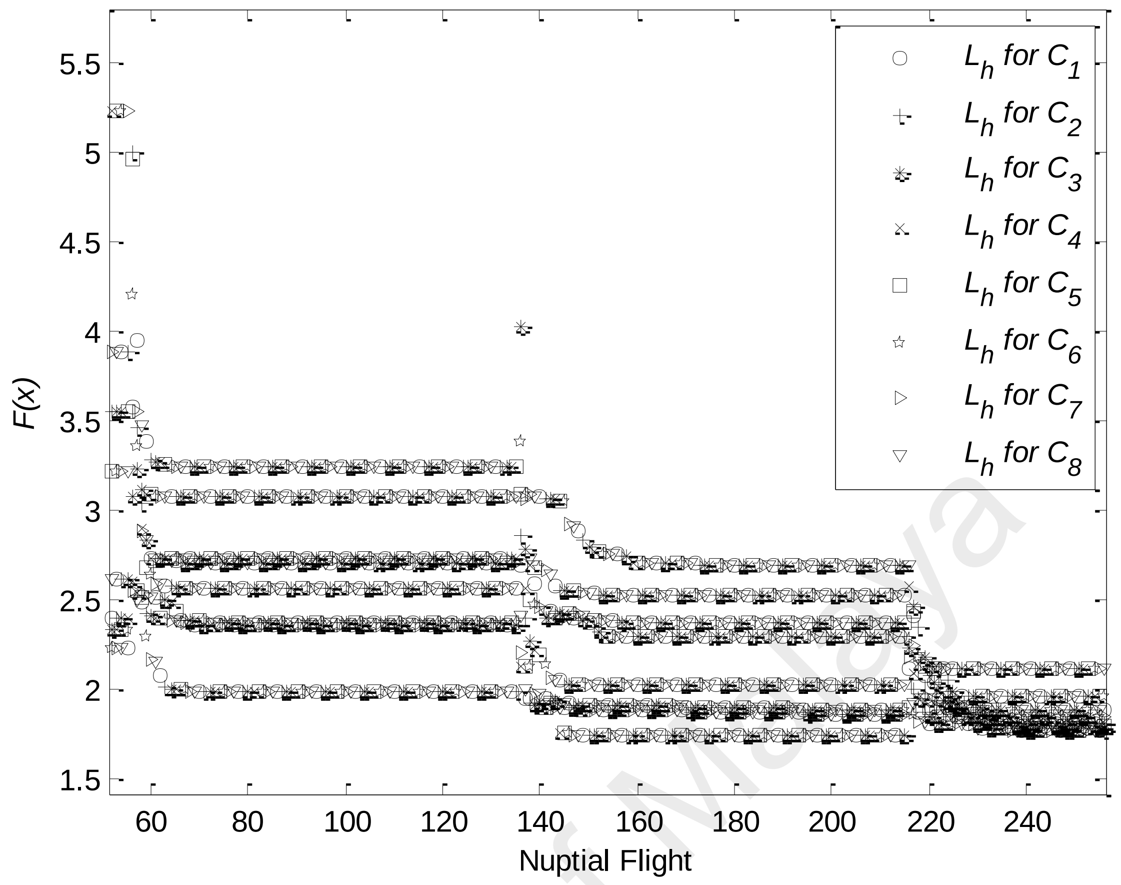

Figure 4.6: Magnified Convergence of Welded Beam Design Problem between nuptial flights 50 to 260

Figure 4.6 shows magnified or enlarged the plot of SDAA solving Welded Beam Problem with the fast saturation can see clearly. This means at the starting plot of each exploration peak, SDAA found nearer to saturation point very fast. This behaviour of SDAA solving Welded Beam Design Problem may give an advantage of low FEs used, but also might have chances to cause SDAA trap in local minima.

4.3 Data Clustering Problem Solving

The SDAA and MSDAA were coded in MATLAB 8.1 (R2012a) running in computer with Windows 7 operating system, 12GB RAM and Intel i7-4770, 3.40GHz processor. The simulations of every data clustering problem were carried out for 30 times. For data clustering optimization, six real data sets from UCI Machine Learning Repository was used to validate SDAA. Each data set has different numbers of clusters, data objects and features as described in the Table 4.8 below (Bache & Lichman, 2013).

Table 4.8: UCI Machine Learning Repository Data Set Information

Dataset	Number of data set, N	Dimension, D	Number of clusters, K
Iris	150	4	3
Wine	178	13	3
Breast Cancer Wisconsin	683	9	2
Contraceptive Method Choice (CMC)	1473	10	3
Glass	214	9	6
Vowel	871	3	6

Table 4.9 to Table 4.14 show the clustering results of different algorithms solving these dataset listed in Table 4.8. By comparing the best and mean result, MSDAA perform very well where it able to achieved minimum result for all the cases. MSDAA also shows that low standard deviation (Std.) which mean not frequent trap on local minima. In other hand, MSDAA have the lowest function evaluation (FEs) compared to all other algorithms in all these problems. It shows great improvement when comparing the FEs of MSDAA and SDAA in Table 4.15.

Table 4.9: Simulation result of Iris dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	113.9865	125.1970	139.7782	14.563	38128
SA	97.4573	99.9570	102.0100	2.018	5314
TS	97.3659	97.8680	98.5694	0.530	20201
ACO	97.1007	97.1715	97.8084	0.367	10998
HBMO	96.7520	96.9531	97.7576	0.531	11214
PSO	96.8942	97.2328	97.8973	0.347	4953
CI	96.6557	96.6561	96.6570	0.0002	7250
K-MCI	**96.6554**	**96.6554**	96.6554	**0**	3500
SDAA	**96.6554**	**96.6554**	**96.6554**	**0**	7080
MSDAA	**96.6554**	**96.6554**	96.6555	**0**	**2800**

Table 4.9 shows the result of solving Iris clustering problem data set. SDAA perform very well to achieve the best result with 0 standard deviation. With the improvement or modification of adding K-means to SDAA, MSDAA sacrificed the extra exploratory search to lower down the FEs. This shows MSDAA have a great improvement on reducing FEs and still able to get the targeted best and mean result. Only little chances getting slightly higher for worst result and standard deviation still considers as 0.

Table 4.10: Simulation result of Wine dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	16530.53	16530.53	16530.53	**0**	33551
SA	16473.48	17521.09	18083.25	753.084	17264
TS	16666.22	16785.45	16837.53	52.073	22716
ACO	16530.53	16530.53	16530.53	**0**	15473
HBMO	16357.28	16357.28	16357.28	**0**	7238
PSO	16345.96	16417.47	16562.31	85.497	16532
CI	16298.01	16300.98	16305.06	2.118	17500
K-MCI	16292.44	16292.70	16292.88	0.130	6250
SDAA	16292.23	**16292.24**	**16292.51**	8.24E-02	10150
MSDAA	**16292.20**	**16292.24**	16292.62	0.1782	**2900**

Table 4.10 shows that the simulation result of the Wine dataset clustering problem. MSDAA shows the lowest best result compare other algorithms. This shows the advantage of K-means search reduces the waste of extra exploration steps of SDAA and

exploit toward the best answer. MSDAA saved a lot of FEs used with the help of K-Means by the sacrificed increase small amount of standard deviation.

Table 4.11: Simulation result of Cancer dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	2999.32	3249.46	3427.43	229.734	20221
SA	2993.45	3239.17	3421.95	230.192	17387
TS	2982.84	3251.37	3434.16	232.217	18981
ACO	2970.49	3046.06	3242.01	90.500	15983
HBMO	2989.94	3112.42	3210.78	103.471	19982
PSO	2973.50	3050.04	3318.88	110.801	16290
CI	2964.64	2964.78	2964.96	0.094	7500
K-MCI	**2964.38**	**2964.38**	**2964.38**	**0**	5000
SDAA	**2964.38**	**2964.38**	**2964.38**	**0**	9860
MSDAA	**2964.38**	**2964.38**	**2964.38**	**0**	**2900**

MSDAA showing excellent results in solving Cancer dataset clustering problem. The Table 4.11 shows the FEs of MSDAA reduced to about 70% FEs of SDAA but still maintain the best result with 0 standard deviation.

Table 4.12: Simulation result of CMC dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	5705.63	5756.59	5812.64	50.369	29483
SA	5849.03	5893.48	5966.94	50.867	26829
TS	5885.06	5993.59	5999.80	40.845	28945
ACO	5701.92	5819.13	5912.43	45.634	20436
HBMO	5699.26	5713.98	5725.35	12.690	19496
PSO	5700.98	5820.96	5923.24	46.959	21456
CI	5695.33	5696.01	5696.89	0.482	30000
K-MCI	**5693.73**	**5693.75**	5693.80	0.014	15000
SDAA	**5693.73**	5694.01	5694.29	0.247	11020
MSDAA	**5693.73**	**5693.75**	**5693.79**	**0.01366**	**2900**

Result in Table 4.12 shows the improvement of SDAA to MSDAA especially on FEs. The mean result also gets closer to the best result after modification to MSDAA. The result of this table able to say that K-means suitable to solve CMC dataset clustering problem where K-MCI and MSDAA both merged with K-means improved the best and mean result.

Table 4.13: Simulation result of Glass dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	278.37	282.32	286.77	4.138	199892
SA	275.16	282.19	287.18	4.238	199438
TS	279.87	283.79	286.47	4.190	199574
ACO	269.72	273.46	280.08	3.584	196581
HBMO	245.73	247.71	249.54	2.438	195439
PSO	270.57	275.71	283.52	4.550	198765
CI	219.37	223.31	225.48	1.766	55000
K-MCI	212.34	212.57	212.80	0.135	25000
SDAA	210.53	220.05	234.33	9.453	11310
MSDAA	**210.47**	**210.50**	**210.52**	**0.0142**	**3000**

Glass dataset clustering problem has the highest number of clusters with high dimensions problems among all the dataset problems. Most of the algorithms traps on local minima. With the help of K-mean merged with SDAA become MSDAA, the best and the mean result outperform compare others algorithms. The FEs used on MSDAA is highly reduced and still able to get the best result out of all the algorithms compared.

Table 4.14: Simulation result of Vowel dataset

Criteria	Best	Mean	Worst	Std.	FEs
GA	149513.73	159153.49	165991.65	3105.544	10548
SA	149370.47	161566.28	165986.42	2847.085	9423
TS	149468.26	162108.53	165996.42	2846.235	9528
ACO	149395.60	159458.14	165939.82	3485.381	8046
HBMO	149201.63	161431.04	165804.67	2746.041	8436
PSO	148976.01	151999.82	158121.18	2881.346	9635
CI	149139.86	149528.56	150468.36	495.059	15000
K-MCI	**148967.24**	148967.55	149048.58	36.086	7500
SDAA	148967.39	148969.63	148973.26	1.792	10150
MSDAA	**148967.24**	**148967.45**	**148970.69**	**1.160**	**2900**

Vowel dataset clustering problem also one of the highest number of cluster problem. K-MCI and MSDAA perform well by getting the best result. MSDAA successfully improve the mean result and getting lower standard deviation compare K-MCI. In addition, MSDAA achieved the lowest FEs over all the clustering problem datasets.

Table 4.15: Overall averages FEs for all algorithms

GA	**SA**	**TS**	**ACO**	**HBMO**	**PSO**	**CI**	**K-MCI**	**SDAA**	**MSDAA**
55303	45942	49990	44586	43634	44605	22041	10375	9928	**2900**

Based on overall average FEs, we can see MSDAA perform well which remain around 2900 FEs. This result means that the cluster size and number of dimension did not affect much to MSDAA. The overall average FEs is revised from 9928 to 2900 by the modification of SDAA to MSDAA resulted about 70% decrement of FEs. The overall average FEs of different kind data clustering algorithms are compared in the Figure 4.7. MSDAA shows that it used the lowest number of FEs for solving all the listed data clustering problems.

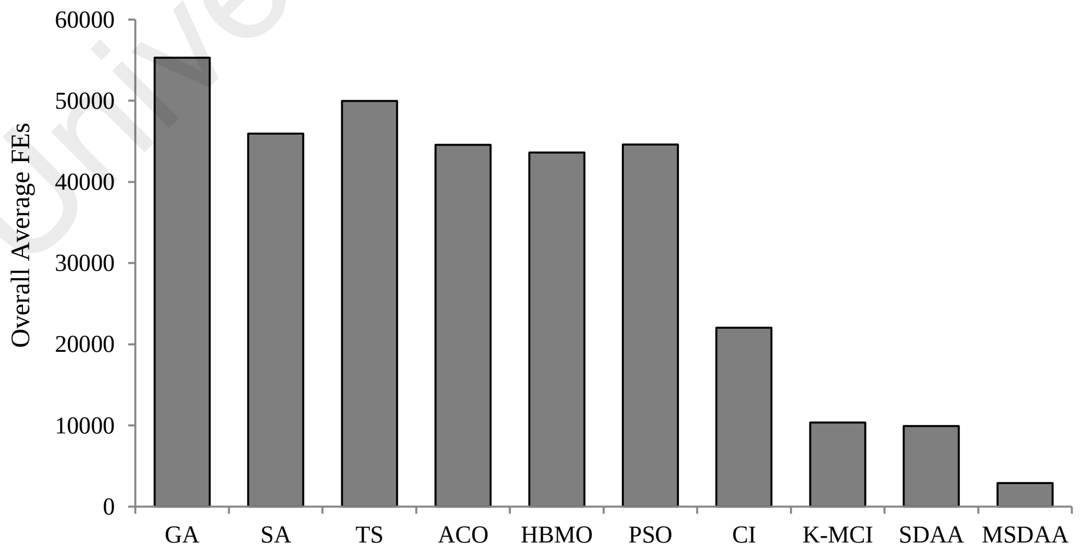

Figure 4.7: Overall average FEs of data clustering algorithms

Table 4.16 to Table 4.20 shows the location of best centroid location found by MSDAA. These are the dimensions or variables found by MSDAA for each cluster centroid to achieve the result in Table 4.9 to Table 4.14.

Table 4.16: Best centroids found by MSDAA for Iris problem (3 clusters)

Dataset	Centroid 1	Centroid 2	Centroid 3
Iris	5.93430	5.01213	6.73329
	2.79785	3.40313	3.06788
	4.41781	1.47164	5.63007
	1.41732	0.23539	2.10681

The Table 4.16 shows the 3 clusters of Iris problem found by MSDAA with every centroid location. There are 4 dimensions value as each centroid as shown in the table.

Table 4.17: Best centroids found by MSDAA for Wine problem (3 clusters)

Dataset	Centroid 1	Centroid 2	Centroid 3
Wine	12.82075	13.73004	12.5031
	2.54474	1.85073	2.32307
	2.37680	2.43719	2.31535
	19.58352	16.91519	21.35024
	98.93214	105.22189	92.56646
	2.06160	2.84618	2.04522
	1.49349	3.05872	1.76859
	0.43120	0.29498	0.39530
	1.42292	2.00281	1.44528
	5.77908	5.69008	4.34390
	0.90115	1.09603	0.93938
	2.19912	3.03242	2.49100
	686.94478	1137.52274	463.53697

Table 4.17 shows the 3 clusters of Wine problem found by MSDAA with every centroid location. There are 13 dimensions value for each centroid as shown in the table.

Table 4.18: Best centroids found by MSDAA for CMC problem (3 clusters)

Dataset	Centroid 1	Centroid 2	Centroid 3
CMC	24.41491	43.63656	33.49713
	3.04027	2.99127	3.13244
	3.50985	3.44247	3.55441
	1.78912	4.59965	3.65207
	0.92444	0.79470	0.78938
	0.79002	0.76585	0.69693
	2.29240	1.82728	2.10167
	2.96867	3.42056	3.28770
	0.03699	0.09194	0.06126
	2.00329	1.67795	2.11263

Table 4.18 shows the 3 clusters of CMC problem found by MSDAA with every centroid location. There are 10 dimensions value for each centroid as shown in the table.

Table 4.19: Best centroids found by MSDAA for Cancer problem (2 clusters)

Dataset	Centroid 1	Centroid 2
Cancer	2.88923	7.11773
	1.12786	6.64029
	1.20049	6.62656
	1.16400	5.61481
	1.99294	5.24096
	1.12158	8.10104
	2.00501	6.07799
	1.10169	6.02385
	1.03127	2.32470

Table 4.19 shows the 2 clusters of Cancer problem found by MSDAA with every centroid location. There are 9 dimensions value for each centroid as shown in the table.

Table 4.20: Best centroids found by MSDAA for Glass problem (6 clusters)

Dataset	Centroid 1	Centroid 2	Centroid 3	Centroid 4	Centroid 5	Centroid 6
Glass	1.52812	1.52001	1.51297	1.51704	1.52103	1.51619
	11.97964	13.25638	13.01245	13.08415	13.73654	14.65215
	0	0.43237	0	3.52666	3.51784	0.05659
	1.08559	1.51943	3.03408	1.36771	1.01966	2.20679
	72.01313	73.02575	70.56388	72.84592	71.89594	73.25357
	0.19669	0.39806	6.21	0.58061	0.21198	0.01067
	14.35597	11.14605	6.98903	8.35731	9.44122	8.68399
	0.15413	0	0	0.01318	0.03471	1.02733
	0.11814	0.06663	0.00143	0.06073	0.05303	0.01929

Table 4.20 shows the 6 clusters of Glass problem found by MSDAA with every centroid location. There are 9 dimensions value for each centroid as shown in the table.

Table 4.21: Best centroids found by MSDAA for Vowel problem (6 clusters)

Dataset	Centroid 1	Centroid 2	Centroid 3	Centroid 4	Centroid 5	Centroid 6
Vowel	506.98293	407.89040	439.24957	357.25663	375.40162	623.71461
	1839.67600	1018.05050	987.68298	2291.44507	2149.42076	1309.59120
	2556.19948	2317.83193	2665.47360	2977.39302	2678.46239	2333.46185

Table 4.20 shows the 6 cluster of Vowel problem found by MSDAA with every centroid location. There are 3 dimensions value for each centroid as shown in the table.

Table 4.22: Parameters used for SDAA and MSDAA for data clustering

SDAA					MSDAA	
C	S	G	r	ε	C	S
10	10	200	0.75	1E-05	10	10

Table 4.21 shows the parameters used by MSDAA compared to SDAA. The parameter used for MSDAA is reduced compared to SDAA.

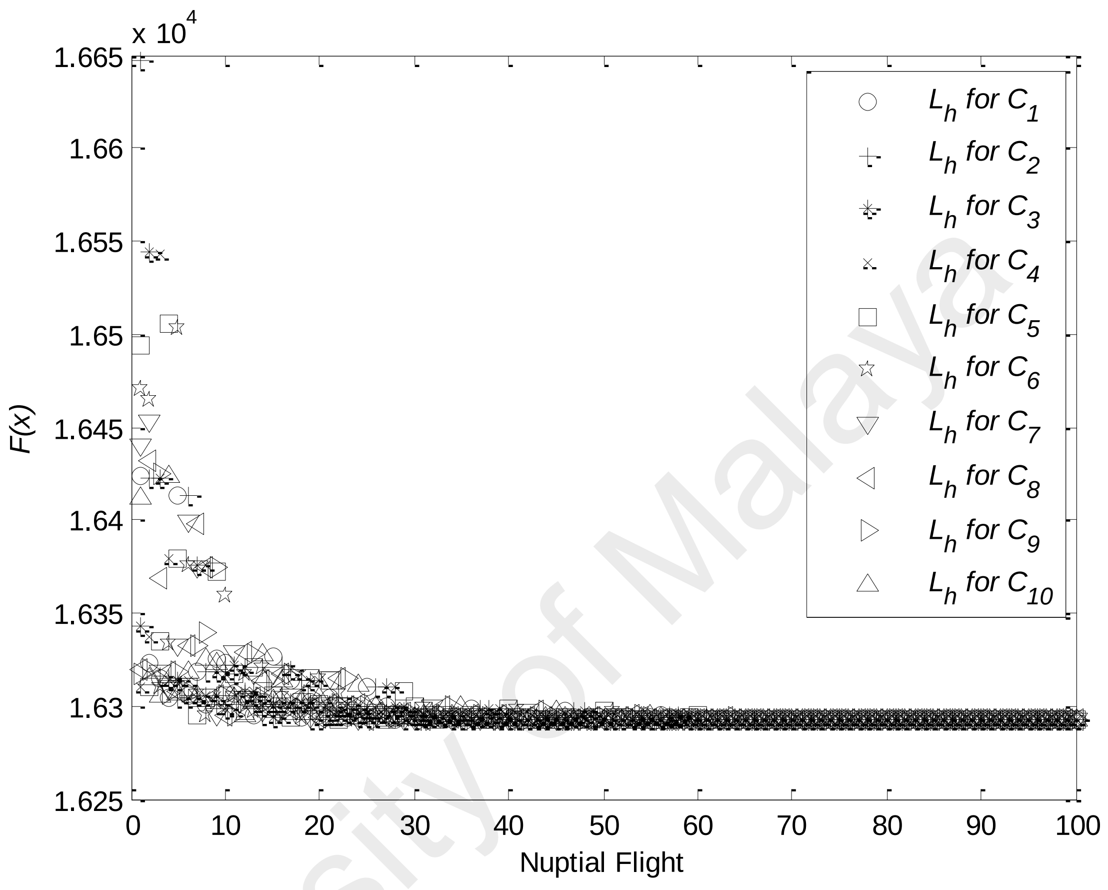

Figure 4.8: Convergence of Wine clustering problem

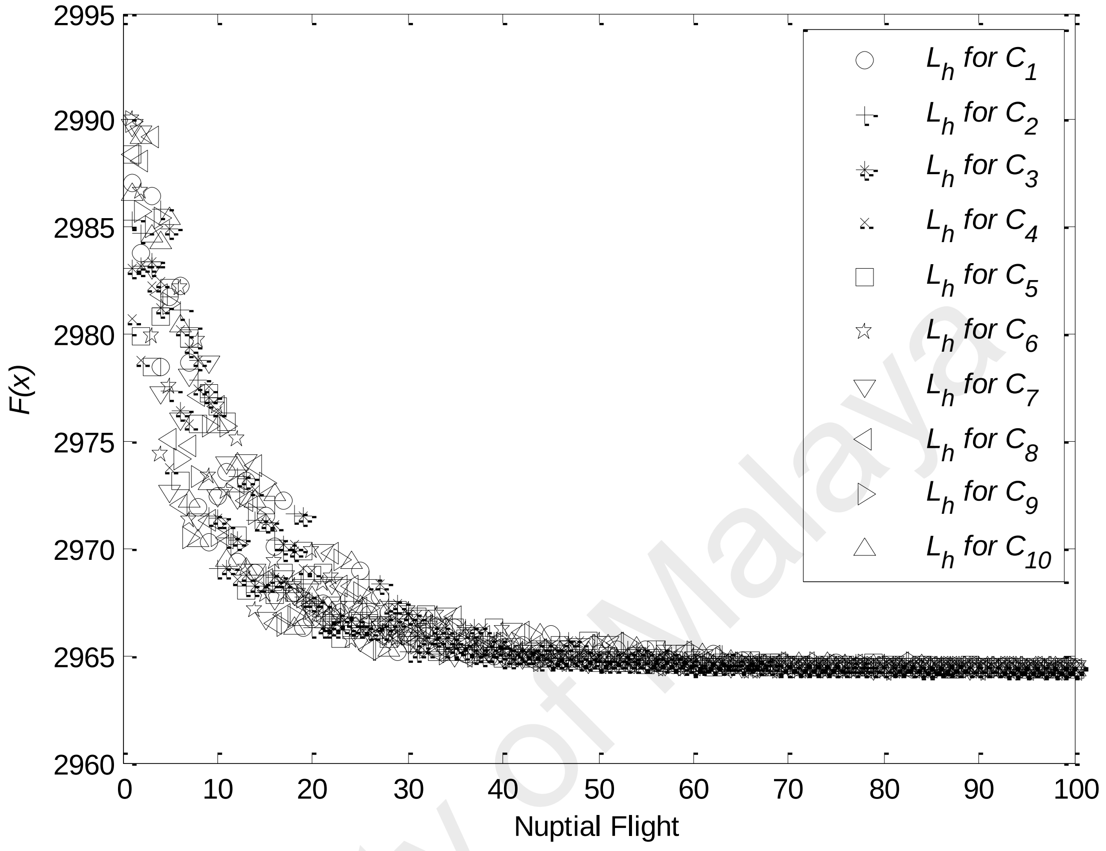

Figure 4.9: Convergence of Cancer clustering problem

Figure 4.8 and Figure 4.9 show convergence plot of Wine and Cancer clustering problems respectively. The convergence plots show the fitness which represented by objective function $f(x)$ versus the number of Nuptial Flight. From the plots, all 10 colonies' male ant L_h mated to others colony's queen in every nuptial flight. The slope of the plot shows that MSDAA able to fast achieve near best result and converge to the global optima.

CHAPTER 5: CONCLUSION

On this research, a new algorithm is developed successfully, namely Seed Disperser Ant Algorithm (SDAA) inspired from the nature ant evolution behaviour. SDAA is applied to different kind of problem solving like real world engineering problems and clustering problems. This shows that SDAA able to solve a variety of problems using SDAA. SDAA also achieve high accuracy and persistency result based on the result shown in Chapter 4. An Improvement applied to SDAA become Modified SDAA (MSDAA) also discussed in this research.

SDAA is successfully created using the ant colony evolution concept where male ants carrying out nuptial flights with queen in other colony to generate young fit queen. The new fit queen will establish a new superior colonies. The optimization process continuously improve is ensured by the nuptial flights process and young queen creation to achieve the global optimum. The new fit queen will establish new superior colonies to explore and help to escape from local optima. SDAA shows comparable result for solving constrained engineering problems and data clustering. However, SDAA still have potential to improve on reducing FEs for data clustering solving. With the purpose of enhancing SDAA, a modified algorithm is created by merging K-means and SDAA name as MSDAA. This advantage of K-means is solving optimization problem using low number of FEs. MSDAA is adopting the advantage of K-means and with the robust search of SDAA. With the behaviour of MSDAA, it show that a great start will reduce the burden of the process to achieve goal. A great algorithm might use a lot of FEs to achieve the goal if the starting point is too far away. K-means in MSDAA successfully reduces a lot of FEs and lead the initial staring point near the global optimum.

REFERENCES

Adorio, E. P., & Diliman, U. (2005). Mvf-multivariate test functions library in c for unconstrained global optimization: Technical report, Department of Mathematics, UP Diliman.

Akay, B., & Karaboga, D. (2012). Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing, 23*(4), 1001-1014.

Al-Sultan, K. S. (1995). A tabu search approach to the clustering problem. *Pattern recognition, 28*(9), 1443-1451.

Ali, L., Sabat, S. L., & Udgata, S. K. (2012). Particle swarm optimisation with stochastic ranking for constrained numerical and engineering benchmark problems. *International Journal of Bio-Inspired Computation, 4*(3), 155-166.

Ashton, M. C., Paunonen, S. V., Helmes, E., & Jackson, D. N. (1998). Kin altruism, reciprocal altruism, and the Big Five personality factors. *Evolution and Human Behavior, 19*(4), 243-255.

Auger, A., & Hansen, N. (2005). *A restart CMA evolution strategy with increasing population size.* Paper presented at the Evolutionary Computation, 2005. The 2005 IEEE Congress on.

Bache, K., & Lichman, M. (2013). UCI machine learning repository. *URL http://archive.ics.uci.edu/ml, 19.*

Back, T. (1996). *Evolutionary algorithms in theory and practice*: Oxford Univ. Press.

Belegundu, A. D., & Arora, J. S. (1985). A study of mathematical programming methods for structural optimization. Part I: Theory. *International Journal for Numerical Methods in Engineering, 21*(9), 1583-1599.

Ben Hadj-Alouane, A., & Bean, J. C. (1997). A genetic algorithm for the multiple-choice integer program. *Operations research, 45*(1), 92-101.

Ben Hamida, S., & Schoenauer, M. (2002). *ASCHEA: new results using adaptive segregational constraint handling.* Paper presented at the Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC'02. .

Blum, C., & Li, X. (2008). *Swarm intelligence in optimization*: Springer.

Chang, W. L., Kanesan, J., & Kulkarni, A. J. (2015). Seed Disperser Ant Algorithm: An Evolutionary Approach for Optimization *Applications of Evolutionary Computation* (pp. 643-654): Springer.

Chang, W. L., Kanesan, J., Kulkarni, A. J., & Ramiah, H. (2017). Data clustering using seed disperser ant algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences*. Advance online publication. doi: 10.3906/elk-1512-23.

Cheron, B., Doums, C., Federici, P., & Monnin, T. (2009). Queen replacement in the monogynous ant< i> Aphaenogaster senilis</i>: supernumerary queens as life insurance. *Animal Behaviour, 78*(6), 1317-1325.

Coello, C. A. C., & Cortés, N. C. (2004). Hybridizing a genetic algorithm with an artificial immune system for global optimization. *Engineering Optimization, 36*(5), 607-634.

Coello Coello, C. A. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*(2), 113-127.

Coello Coello, C. A., & Becerra, R. L. (2004). Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization, 36*(2), 219-236.

Coello Coello, C. A., & Mezura Montes, E. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics, 16*(3), 193-203.

Dorigo, M., & Birattari, M. (2010). Ant colony optimization *Encyclopedia of Machine Learning* (pp. 36-39): Springer.

Eberhart, R. C., & Kennedy, J. (1995). *A new optimizer using particle swarm theory.* Paper presented at the Proceedings of the sixth international symposium on micro machine and human science.

Fathian, M., & Amiri, B. (2008). A honeybee-mating approach for cluster analysis. *The International Journal of Advanced Manufacturing Technology, 38*(7-8), 809-821.

Fathian, M., Amiri, B., & Maroosi, A. (2007). Application of honey-bee mating optimization algorithm on clustering. *Applied Mathematics and Computation, 190*(2), 1502-1513.

Floudas, C. A., & Pardalos, P. M. (1990). *A collection of test problems for constrained global optimization algorithms* (Vol. 455): Springer.

Goldberg, D. E., Zakrzewski, K., Chang, C., & Gallego, P. (1997). Genetic algorithms: A bibliography. *Urbana, 51*, 61801.

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems, 47*, 98-115.

He, Q., & Wang, L. (2007a). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence, 20*(1), 89-99.

He, Q., & Wang, L. (2007b). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation, 186*(2), 1407-1422.

Huang, F.-z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation, 186*(1), 340-356.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters, 31*(8), 651-666.

Jamian, J. J., Abdullah, M. N., Mokhlis, H., Mustafa, M. W., & Bakar, A. H. A. (2014). Global Particle Swarm Optimization for High Dimension Numerical Functions Analysis. *Journal of Applied Mathematics, 2014*.

Kannan, B., & Kramer, S. N. (1994). An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of mechanical design, 116*(2), 405-411.

Kenne, M., & Dejean, A. (1998). Nuptial Flight of Myrmicaria opaciventris. *Sociobiology, 31*(1).

Kennedy, J. (2010). Particle swarm optimization *Encyclopedia of Machine Learning* (pp. 760-766): Springer.

Kennedy, J., & Eberhart, R. (1995, Nov/Dec 1995). *Particle swarm optimization.* Paper presented at the Neural Networks, 1995. Proceedings., IEEE International Conference on.

Koziel, S., & Michalewicz, Z. (1999). Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation, 7*(1), 19-44.

Krishnasamy, G., Kulkarni, A. J., & Paramesran, R. (2014). A hybrid approach for data clustering based on modified cohort intelligence and K-means. *Expert Systems with Applications, 41*(13), 6009-6016.

Kulkarni, A. J., Durugkar, I. P., & Kumar, M. (2013). *Cohort intelligence: a self supervised learning behavior.* Paper presented at the Conference on Systems, Man, and Cybernetics (SMC), 2013 IEEE International

Kulkarni, A. J., & Tai, K. (2009). Probability collectives: a decentralized, distributed optimization for multi-agent systems *Applications of Soft Computing* (pp. 441-450): Springer.

Lampinen, J. (2002). *A constraint handling approach for the differential evolution algorithm.* Paper presented at the Computational Intelligence, Proceedings of the World on Congress on.

Lee, T.-Y., & Chen, C.-L. (2007). Unit commitment with probabilistic reserve: An IPSO approach. *Energy conversion and Management, 48*(2), 486-493.

Liang, J., Suganthan, P., & Deb, K. (2005). *Novel composition test functions for numerical global optimization.* Paper presented at the Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*(2), 629-640.

Liu, L., Zhong, W.-M., & Qian, F. (2010). An improved chaos-particle swarm optimization algorithm. *Journal of East China University of Science and Technology, 36*(2), 267-272.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern recognition, 33*(9), 1455-1465.

Mezura-Montes, E., & Coello, C. A. C. (2005). Useful infeasible solutions in engineering optimization with evolutionary algorithms *MICAI 2005: Advances in Artificial Intelligence* (pp. 652-662): Springer.

Michalewicz, Z., & Attia, N. (1994). *Evolutionary optimization of constrained problems.* Paper presented at the Proceedings of the 3rd annual conference on evolutionary programming.

Miranda, V., & Fonseca, N. (2002). *EPSO-evolutionary particle swarm optimization, a new algorithm with applications in power systems.* Paper presented at the Proc. of the Asia Pacific IEEE/PES Transmission and Distribution Conference and Exhibition.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46-61.

Mitchell, M. (1998). *An introduction to genetic algorithms*: MIT press.

Molga, M., & Smutnicki, C. (2005). Test functions for optimization needs. *Test functions for optimization needs*.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report, 826*, 1989.

Moscato, P., Cotta, C., & Mendes, A. (2004). Memetic algorithms *New optimization techniques in engineering* (pp. 53-85): Springer.

Mozaffari, A., Gorji-Bandpy, M., & Gorji, T. B. (2012). Optimal design of constraint engineering systems: application of mutable smart bee algorithm. *International Journal of Bio-Inspired Computation, 4*(3), 167-180.

Niknam, T., & Amiri, B. (2010). An efficient hybrid approach based on PSO, ACO and< i> k</i>-means for cluster analysis. *Applied Soft Computing, 10*(1), 183-197.

Osiński, J. (2009). Kin altruism, reciprocal altruism and social discounting. *Personality and Individual Differences, 47*(4), 374-378.

Pan, G., Li, K., Ouyang, A., Zhou, X., & Xu, Y. (2014). A hybrid clustering algorithm combining cloud model iwo and k-means. *International Journal of Pattern Recognition and Artificial Intelligence, 28*(06), 1450015.

Pei, S., Ouyang, A., & Tong, L. (2012). A Hybrid Algorithm Based on Bat-Inspired Algorithm and Differential Evolution for Constrained Optimization Problems. *International Journal of Pattern Recognition and Artificial Intelligence*, 1559007.

Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *Evolutionary Computation, IEEE Transactions on, 13*(2), 398-417.

Ramezani, P., Ahangaran, M., & Yang, X.-S. (2013). Constrained optimisation and robust function optimisation with EIWO. *International Journal of Bio-Inspired Computation, 5*(2), 84-98.

Rao, S. S., & Rao, S. (2009). *Engineering optimization: theory and practice*: John Wiley & Sons.

Selim, S. Z., & Alsultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern recognition, 24*(10), 1003-1008.

Shang, Y.-W., & Qiu, Y.-H. (2006). A note on the extended Rosenbrock function. *Evolutionary Computation, 14*(1), 119-126.

Shelokar, P., Jayaraman, V. K., & Kulkarni, B. D. (2004). An ant colony approach for clustering. *Analytica Chimica Acta, 509*(2), 187-195.

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.

Tsai, C.-Y., & Kao, I.-W. (2011). Particle swarm optimization with selective particle regeneration for data clustering. *Expert Systems with Applications, 38*(6), 6565-6576.

Wang, L., & Li, L.-p. (2010). An effective differential evolution with level comparison for constrained engineering design. *Structural and Multidisciplinary Optimization, 41*(6), 947-963.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on, 1*(1), 67-82.

Xu, W., Geng, Z., Zhu, Q., & Gu, X. (2013). A piecewise linear chaotic map and sequential quadratic programming based robust hybrid particle swarm optimization. *Information Sciences, 218*, 85-102.

Yang, C., Zhang, X., Zhong, C., Liu, C., Pei, J., Ramamohanarao, K., & Chen, J. (2014). A spatiotemporal compression based approach for efficient big data processing on cloud. *Journal of Computer and System Sciences*.

Yang, X.-S., & Deb, S. (2009). *Cuckoo search via Lévy flights.* Paper presented at the Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on.

Yang, X.-S., Huyck, C., Karamanoglu, M., & Khan, N. (2013). True global optimality of the pressure vessel design problem: a benchmark for bio-inspired optimisation algorithms. *International Journal of Bio-Inspired Computation, 5*(6), 329-335.

Yıldız, A. R. (2009). A novel particle swarm optimization approach for product design and manufacturing. *The International Journal of Advanced Manufacturing Technology, 40*(5-6), 617-628.

Yoo, J., & Hajela, P. (1999). Immune network simulations in multicriterion design. *Structural Optimization, 18*(2-3), 85-94.

Zahara, E., & Kao, Y.-T. (2009). Hybrid Nelder–Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications, 36*(2), 3880-3886.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Conference paper:

Chang, W. L., Kanesan, J., & Kulkarni, A. J. (2015). Seed Disperser Ant Algorithm: An Evolutionary Approach for Optimization *Applications of Evolutionary Computation* (pp. 643-654): Springer.

Journal article:

Chang, W. L., Kanesan, J., Kulkarni, A. J., & Ramiah, H. (2017). Data clustering using seed disperser ant algorithm. *Turkish Journal of Electrical Engineering & Computer Sciences*. Advance online publication. doi: 10.3906/elk-1512-23.