CERTIFICATION OF APPROVAL

Application of Bee Colony Optimization (BCO) in NP-Hard Problems

by

Muhammad Sariy Syazwan Bin Kamarudin

A project dissertation submitted to the
Computer and Information Science Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR (Hons) BUSINESS INFORMATION SYSTEM

Approved by,

[illegible]

(Mr. Helmi Bin Md Rais)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

[illegible]

MUHAMMAD SARIY SYAZWAN B. KAMARUDIN

ACKNOWLEDGEMENT

First and foremost, I would like to express my heartiest gratitude to Allah the Almighty, for that without His grace and mercy, I would not be able to make it through this journey until the very end line.

I also would like to express my utmost appreciation to my Supervisor, Mr. Helmi Bin Mohd Rais, Lecturer of Universiti Teknologi PETRONAS for the great support, encouragement, guidance and the trust he had given me. Without whom, I will not be able to gain as much knowledge and success as I had been able now. I would also like to show my appreciation to the Computer and Information Science Head of Department, Dr. Fadzil for all the kind assistance in giving me the opportunity to do this research that might be helping to equip me as to be a good pick for the industry in the future. The supervision throughout the industrial internship period was such a great assistance and was very much appreciated.

My sincere gratitude also goes to Ms. Rohmah, lecturer of Universiti Teknologi PETRONAS and our instructor for Final Year Project and also other dedicated personnel for managing and facilitating us in completing this Final Year Project.

Million thanks are especially dedicated to all my friends for being very supportive and helpful. Million thanks to the both parents, Mr. Kamarudin Bin Mamat and Mrs. Ruhana Binti Yaacob for their greatest support and trust they have in me, for the prayers to ease my journey.

Thank you.

ABSTRACT

This report presents the first stage of an ongoing research which is the problem solving of highly complex tasks using Bee-Inspired algorithms. Bee-Inspired algorithms are partially referring to the nature of bee swarm behaviour. Bee swarm behaviour characterized by autonomy and distributed functioning and self-organizing. Nowadays, lots algorithms under the field of Swarm Intelligent have been used to solve complex problem. Various Artificial Intelligence systems have been developed based on plenty of researches and studies done on the behaviour of social insects. Bee-Inspired algorithms were presumed to bring the new direction in the field of Swarm Intelligence.

The primary goal of this research is to prove the efficiency of collective bee intelligence in solving problems characterized by uncertainty. P-Median problem will be used as the main case study in implementing one of the Bee-Inspired algorithms to prove, illustrate example and show the characteristics of the selected concept.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATION AND NOMENCLATURES

Term	Abbreviation	Definition
Travelling Salesman Problem	TSP	An optimization problem under NP-Hard category
Non-deterministic Polynomial-time Hard	NP-Hard	One of the category for optimization problem
Ant Colony System	ACS	An ant-inspired algorithm
P-Median Problem	PMP	An optimization problem under NP-Hard category
BS	Bee System	A bee-inspired algorithm
BeeHive Algorithm	BHA	A bee-inspired algorithm
BeeHiveGuard	BHG	Extended version of BHA
BeeHiveArtificial Iintelligence System	BHAIS	Extended version of BHA
Honey Bee Algorithm	HBA	A bee-inspired algorithm
Virtual Bee Algorithm	VBA	A bee-inspired algorithm
Honey-bee Mating Optimization	HBMO	A bee-inspired algorithm
BeeAdHoc Model	BAHM	A bee-inspired algorithm
Bee Colony Optimization	BCO	A bee-inspired algorithm
Bee Swarm Optimization	BSO	A bee-inspired algorithm
Artificial Bee Colony	ABC	A bee-inspired algorithm
Honey Bee Social Foraging	HBSF	A bee-inspired algorithm
Frequency Based Pruning Strategy	FBPS	Fitness calculation
Simulated Bee Colony	SBC	A bee-inspired algorithm

CHAPTER 1: INTRODUCTION

In this chapter, we will discuss thoroughly about the overview of this ongoing research. Basically it will include the following topics:

- Section 1.1 – BACKGROUND of STUDY
- Section 1.2 – PROBLEM STATEMENT
- Section 1.3 – OBJECTIVES and SCOPE of STUDY
- Section 1.4 – FEASIBILITY of THE PROJECT
- Section 1.5 – EXPECTED OUTCOMES
- Section 1.6 – CHALLENGES

1.1 BACKGROUND of STUDY

Since long time ago, people have noticed the wonderful characteristics and behaviour of insects and animal in this world. A group of bees forage honey in groups. A flock of birds move in group forming a quite specific shape in the sky. A group of ants works together in collecting foods. A school of fish move and swims together and etc. Those kind of behaviour was noticed as a natural form of aggregate motion which later been called "Swarm Behaviour". Recently biologists and computer scientists in the field of "artificial life" have studied how to model biological swarms to understand how such "social animals" interact, achieve goals and evolve [12].

Those researches have attracted most engineers in Artificial Intelligence field to further discover that kind of swarm behaviour to be applied as "swarm intelligence". Swarm intelligence were said to be very helpful in solving some complex problems regarding optimization of lots cases mostly concerning logistics and transportation matters. Nowadays, most swarm intelligence also has been applied to robotics (Swarm Robotic), telecommunication, military and others.

This paper will focus on one of the swarm behaviour which is bees' natural foraging behaviour. Travelling Salesman Problem and P-Median Problem will be used as the main problems in analysing bee-inspired algorithm in this paper.

1.2 PROBLEM STATEMENT

1.2.1 Travelling Salesman Problem (TSP)

The Travelling Salesman Problem is an NP-hard (non-deterministic polynomial-time hard) problem in combinatorial optimization studied in operations research and theoretical computer science [11]. In simplest term, TSP can be explained as a way to find the shortest path in a given list of cities (nodes) and the pairwise of their distances. The problem was recognized as mathematical problem in early 1930 while studies in optimization problems were being focused thoroughly. This problem was first intensively been looked into in early 1800s by two renowned mathematicians, Sir William Rowan Hamilton of the Irish and by Thomas Penyngton Kirkman, a British Mathematician. TSP commonly described as a graph problem where nodes were referred as the coordinate while the shortest distance between each nodes were calculated.

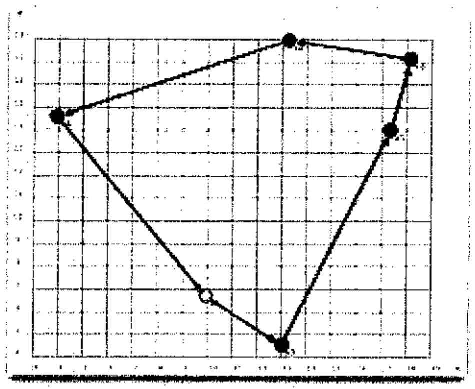


Figure 1: TSP as graph problem

TSP was found in lots cases especially concerning transportation and logistics matter. In many applications, additional restrictions or constraints such as time windows or limited resources make the problem significantly harder.

One of selected bee-inspired will be applied to this problem and the optimized result will be compare to Ant Colony System (ACS) algorithm. The result of the comparison will be shown in statistic at the end of this project.

1.2.1 P-Median Problem (PMP)

The p-Median Problem (PMP) classified as minisum location-allocation problems. These problems find medians among existing points, which is not the same as finding centers among points, a characteristic of minimax location–allocation problems (the p-center problem is an example, where the goal is to minimize the maximum distance between points and center) [J.Reese, 2006]. In a simple description for understanding PMP, Bader F. AlBdaiwi ,Diptesh Ghosh and Boris Goldengorin defined it as below:

> Given a set *I* of *m* potential facilities, a set *J* of *n* users (or customers), a distance function $c: I \times J \rightarrow \Re_{+}$, and a constant $p \le m$, determine which p facilities to open so as to minimize the sum of the distances from each user to its closest open facility.

1.3 OBJECTIVES and SCOPE of STUDY

1.3.1 Objectives

The three main objectives of this research are:

- To do research on different kinds of bee-inspired algorithm
- To implement and make comparison between one of the bee-inspired algorithm and ACS algorithm to solve Travelling Salesman Problem
- To apply one of the bee-inspired algorithm to solve P-Median problem

1.3.2 Scope of Study

This project paper was mainly developed to study about the bee inspired algorithms. The findings about bee-inspired algorithms will include the founder, development year and the problem that suits the algorithms the best. The study also will include the implementation of the chosen bee-inspired algorithm to solve two NP-Hard problems which are TSP and PMP. The project will also include comparison between the chosen bee-inspired algorithm and ACS in solving TSP problem.

1.4 FEASIBILITY of THE PROJECT

1.4.1 Scope Feasibility

This project paper will focus mainly on implementing the chosen bee-inspired algorithm on the PMP problem. The system will import the data set from notepad and execute it to find the median of facilities that would serve all the client nodes the best. This paper also will generate comparison between the results produced by bee-inspired algorithm and ACS in solving TSP problem. The same data set will be used during the execution of both algorithms to apply the purpose of fair comparison. Last but not least, this paper will also include the study done on almost all bee-inspired algorithms that was developed up until now. The study will include the findings of the developer name(s), year of development and the application that best served by the algorithm.

1.4.2 Schedule Feasibility

The project is expected to be completed within the 8 month budgeted time frame given. All the project objectives are expected to be met in parallel upon the end of the development stage. One third of the development phase will be done in parallel with testing phase to produce the comparison in results of bee-inspired algorithm and ACS in solving TSP problem.

1.4.3 Technical Feasibility

The writer has the basic knowledge in C++ programming language which could be a help in completing the project. The basic knowledge allows the author to have the clearer picture of the algorithm. Some additional knowledge might be required but it is not a big problem as the author already familiar with the development language. The supervisor is also an expert in the C++ programming language which gives an advantage for the author to learn more about C++ in completing the project.

1.5 EXPECTED OUTCOMES

The project was expected to result in enough and efficient findings about almost all bee-inspired algorithms in the end of the project. This finding could bring clearer picture about the evolvement of bee-inspired algorithms. Hopefully, the findings could lead to generation of new idea in the area of swarm intelligence which could be a help for lot cases happen in the world.

It also was foreseen to successfully prove that the chosen bee-inspired algorithm is better than ACS in solving TSP problem. This will be presented in statistical data to make comparison of optimized solutions done by both algorithms. This result could bring a new direction in solving complex problem related to traffic pattern in transportation problems.

The last but not least expected outcome is the successfulness of applying the chosen bee-inspired algorithm to the PMP problem. This will be the great achievement of the project as it will include a lot of challenges in making the outcome a reality. This outcome also could strengthen other researches view about the ability of bee-inspired algorithms in solving complex problems in our daily life.

1.6 CHALLENGES

1.6.1 Limited Access to Original Sources

There are a lot of study need to be conducted in order to complete the research of this project. It cannot be simply done without reffering to the correct or original sources as it could lead to the failure of the project. Lots original sources that was developed within this project scope were not published and were claimed as the individual intellectual properties. Some sources were for sale and it was hard to get those books or journal as lots researches were done not within Malaysia. This problem tend to bring some flaws in the findings. However, internet still be a usefull help as the main source of references for this project research.

1.6.2 Complex Algorithm

Bee-inspired algorithms were really complex to be understood in a short period of time. Thorough analysis and hands-on work have to be done in order to capture the basic concept of a bee-inspired algorithm. The algorithms involve so many equations and concepts that were confusingly translated by some researches. Analysis that was done based on many resources could be more difficult as the authors might have different views in defining the algorithms. Simply said, the theory was hardly to be understood by only readings. It should be simpler to understand by practicing and reviewing the codes of the algorithms themselves.

1.6.3 Complex understanding of C++ is required

To apply or understand the basic concept of the algorithms could be harder without thorough knowledge in C++ development language. The coding was developed in C++ language and was complicated for beginners to understand the whole coding. Further knowledge in C++ is needed to successfully complete this project.

CHAPTER 2: LITERATURE REVIEW

This literature review mainly created to give an overview of Bee-Inspired algorithms and the proposed solution for TSP and P-Median problem. This part is organized in the following way:

- Section 2.1 – BEE-INSPIRED ALGORITHMS
- Section 2.2 – THE CHOSEN ALGORITHM
- Section 2.3 – BCO in TSP
- Section 2.4 – BCO in PMP

2.1 BEE-INSPIRED ALGORITHMS

This section will briefly discuss about variety of Bee-Inspired Algorithms that have been studied for a decade. A Bee-Inspired Algorithm is not perfect. Some algorithms might be best for a case study but might give less impact on other problems. This section will point out all findings of Bee-Inspired Algorithms and the problems that suit them the best.

2.1.1 Bee System (BS)

Studied done by Sato and Hagiwara based on the honey bee behaviour has come out with a model called Bee System in 1997. The model has been added with the behaviour of scout bees to introduce new potential solutions and avoid premature convergence to local minima solution. This model has been used by Lucic and Teodorovic in 2002 to be applied on solving complex traffic and transportation problem. The model was successfully applied to solve eight benchmark version of Travelling Salesman Problem.

2.1.2 BeeHive Algorithm (BHA)

In 2004, Wedde et al. has developed a BeeHive algorithm which has been inspired by the communication in the hive of honey bees and they applied the BeeHive algorithm to the routing in networks [10]. The BHA was extended to BeeHiveGuard (BHG) and BeeHiveAIS (BHAIS) to provide a model for countering security threats from BHA.

2.1.3 Honey Bee Algorithm (HBA)

Honey Bee Algorithm (HBA) has been formulated by Nakrani and Tovey in 2004 paper. Their model was focusing on the problem of dynamic allocation on internet services. It is the problem regarding the matter of allocating computers among different users and web-hosting server on the internet.

2.1.4 Virtual Bee Algorithm (VBA)

In the late 2004 and early 2005, Yang from Cambridge University has developed an algorithm called Virtual Bee Algorithm (VBA) to solve numerical optimization problem. There are only two parameters were given in the study but it seems that the algorithm could handle and optimize both function and discrete problems. The author claims that VBA have some similarities with Particle Swarm Optimization (PSO) in certain parts.

2.1.5 Honey-bee Mating Optimization (HBMO)

A bit later after the development of VBA in 2005, Haddad and Afshar and their other members have presented HBMO in the same year. Their studied was about applying HBMO in reservoir modelling and clustering.

2.1.6 BeeAdHoc Model (BAHM)

BeeAdHoc is an on-demand multi path routing algorithm for mobile adhoc networks inspired from the foraging principles of honey bees [7]. This study once again been done by Wedde and Farooq in 2005. The purpose of the study is to find an energy efficient routing method to be applied in mobile ad hoc network. Mobile Ad HOC networks (MANET'S) are networks in which all nodes are mobile and communicate with each other via wireless connections [7].

2.1.7 Bee Colony Optimization (BCO) Metaheuristic

Inspired by BS algorithm, Teodorovic and Dell came out with new model called BCO metaheuristic in 2005. The study was successfully done by applying the model to solve ride-matching problem. In 2006, extended study has been done by Chong et al. on the same model (BCO). This study was done base on dance duration of honey bee to select a path. This model has been applied to job scheduling problem in the same year.

2.1.8 Bee Swarm Optimization (BSO)

A studied in 2005 by Drias and Yahi using meta-heuristic model called Bee Swarm Optimization has been conclude to outperform other swarm inspired algorithms in solving Maximum Satisfiability problem. They specifically referred the other swarm inspired algorithms as ant colony algorithm as their research has been comparing the result with that algorithm. In the same studied, Drias and Yahi also successfully applied their model as a solution for MAX-W-SAT problem.

2.1.9 Artificial Bee Colony (ABC)

First study on ABC was done by Karaboga in 2005. The study was initially done focusing on solving numerical optimization. The same study was extended by Basturk and Karaboga in 2006 to solve multi-dimensional numerical problem. They conclude that the use of ABC in solving those multi-dimensional problems produced the results which are at par with solutions done by differential evolution, particle swarm optimization and evolutionary algorithm. The study on ABC later was extended in 2008 and 2009 by Rao et al., Karaboga and Singh for solving integer programming and engineering design problem. In 2009, Karaboga and Ozturk also have extended the use of ABC for neural network training. Latest extension versions of ABC have been done recently in 2010 by Pan et al., Omkar et al., Karaboga, Ozturk, Xu and Duan on for problems like combinatorial solutions, multi-object optimization problem, clustering solution, and image processing.

2.1.10 Honey Bee Social Foraging (HBSF)

Honey Bee Social Foraging was first been studied by Quijino and Passino in 2007. Their study was focusing on optimizing the behaviour of foraging bees to solve a class of optimal resource allocation problem. The model has been successfully applied in a multizone temperature control grid, where the control objective is to seek the maximum uniform temperature [8]. The test has successfully visualized dynamic re-allocation, cross-inhibition and the ideal free distribution.

2.1.11 Simulated Bee Colony (SBC)

The study on SBC model was done by McCaffrey in 2009. The study was about demonstrating SBC in solving problem of generating pairwise set. The study was concluded to successfully outperform existing deterministic algorithm for six out of seven benchmark set of the problem.

2.2 THE CHOSEN ALGORITHM

In this project, Bee Colony Optimization (BCO) metaheuristic algorithm has been chosen to be implemented in TSP and PMP problem. As mentioned above (see 2.1.7), BCO metaheuristic was first proposed in Lucic and Teodorovic in 2005. The proposed algorithm was referred as a meta-heuristic algorithm due to its capability in solving various optimization problems in management, engineering, and control. The way BCO been developed was partially according to the way in which bees in nature looking for food. Naturally, scout bees follow a certain steps in their way to explore the sources of pollen, nectar and propolis (their food sources):

1. Scout-Bees in nature leave the hive and explore the areas within the certain distance from their hive.
2. Go back to their hive and exchange information about the locations, quantity, and quality of their sources of food. They exchange information through a ritual called "waggle dance".
3. Using information given, foraging bees will follow one of the scout bees to the locations of the food. Upon arrival to the food source, the foraging bees will be collecting as much food as they can and go back to their hive to store the food.
4. After all food been stored, there are 3 possible decisions will be made by the foraging bees:
    - The bee can discard the food location and become again an uncommitted follower.
    - It can keep on with the foraging behaviour at the discovered nectar source, with no recruiting the rest of the colony.
    - It can recruit other bees with the dance ritual prior to return to the food location.

The decision where made upon the quality, quantity and the distance of the food sources.

2.3 BCO in TSP

As per discussed above (see 1.2.1), TSP could be defined as a problem of finding the shortest distance that start from a node, move through other nodes and come back again to the starting node. BCO metaheuristic algorithm will randomly select one of the nodes as the starting point and decompose the problem into stages.

During each stages (forward pass stages), bees will calculate new distance among the nodes to be added to the partial solution of TSP tour. The selection of new starting node will be randomly done using certain probabilities. The probabilities of choosing the next node to be visited were commonly done using Logit-Based Model. However, in this research, we will try to use the Frequency Based Pruning Strategy (FBPS). FBPS suggests that only a subset of promising solutions are allowed to perform 2-opt based on the accumulated frequency of its building blocks recorded in a matrix [14]. The basic understanding of the strategy is that BCO will run the operation based on calculated finesses done in matrix using Building Block Concept. This concept actually almost the same concept used in Ant Colony Optimization which is called pheromones.

Let us take example of 6 cities in TSP problem to better understand the Building Block Concept. First of all, a 6 x 6 matrix will be created and assigned to 0:

	A	B	C	D	E	F
A	0	0	0	0	0	0
B	0	0	0	0	0	0
C	0	0	0	0	0	0
D	0	0	0	0	0	0
E	0	0	0	0	0	0
F	0	0	0	0	0	0

Table 1: Building Block Matrix (Initialization)

If let say we found solution of "A, B, C, D, E, F, A", the building block will be updated as follow:

	A	B	C	D	E	F
A	0	1	0	0	0	1
B	1	0	1	0	0	0
C	0	1	0	1	0	0
D	0	0	1	0	1	0
E	0	0	0	1	0	1
F	1	0	0	0	1	0

Table 2: Building Block Matrix (First Cycle)

For 1 pair of solution for example, AB, both matrix for AB and BA will be updated and if we assume the next solution will be, "A, B, E, D, C, F, A" blocks below will be updated:

	A	B	C	D	E	F
A	0	2	0	0	0	2
B	2	0	1	0	1	0
C	0	1	0	2	0	1
D	0	0	2	0	2	0
E	0	1	0	2	0	1
F	2	0	1	0	1	0

Table 3: Building Block Update (Second Cycle)

After several iterations, we can assume to have a matrix of fitness as below:

	A	B	C	D	E	F
A	0	88	929	22	113	23
B	88	0	754	355	105	4
C	929	754	0	11	826	2
D	22	355	11	0	176	933
E	113	105	826	176	0	56
F	23	4	2	933	56	0

Table 4: Building Block Update (Several Cycles)

From the gathered entries of the matrix, we can calculate the fitness percentage of every pair. For example, pair for A can be AB, AC, AD, AE and AF. Total amount of fitness of all pair for A will be 1175 (88+929+22+113+23). So, the percentage fitness can be calculated for example pair AB has a percentage fitness of 7.49% (88/1175x100). So, the table above can be updated as follow:

	A	B	C	D	E	F
A	0	7.49	79.06	1.87	9.62	1.96
B	6.74	0	57.73	27.18	8.04	0.31
C	36.84	29.90	0	0.44	32.75	0.08
D	1.47	23.71	0.73	0	11.76	62.32
E	8.86	8.23	64.73	13.79	0	4.39
F	2.26	0.39	0.20	91.65	5.50	0

Table 5: Pair Fitness Percentage

Using the fitness matrix, BCO will run the process of finding the best solution based on the fitness percentage of each pair. It means that pair with higher percentage will have more chances to be chosen in the calculation process of BCO algorithm.

After all partial solutions been gathered, the best solutions will be used to update the global best solution. The whole process will represent an end of a single iteration. The next iteration will begin by choosing a new starting point until the conditions are met.

2.4 BCO in PMP

PMP was known to be first recognized by Hakimi in 1964 to 1965. This NP-Hard problem can be defined as to find the location of facilities on a network so that the summation of all demand nodes is closest to those locations.

In this problem, starting node will be place at the located median or the facility node. The next stage is to select the next node to be chosen and added to the partial solution. In selecting the next node, we will have to use the concept of bees' utility. The utility V_i in this case were used to demonstrate the calculation if the i node is chosen as the starting point.

$$V_i = wR_i + qA_i\,, i = 1,....,n; \qquad (1)$$

Where,

R_i - normalized value of the distance from the starting node to the i-th node.

A_i - normalized value of demand in the i-th node

w, q - weight of the distance and demand

R_i and A_i can be calculated as below:

$$R_i = \frac{r_{max} - r_i}{r_{max} - r_{min}}\,,\ A_i = \frac{a_i - a_{min}}{a_{max} - a_{min}}\,,\ R_i\,, A_i \in [0,1]\,, i = 1,....,n; \quad (2)$$

Where,

r_i - the total distance from the starting node to the i-th node.

r_{max} , r_{min} - minimum and maximum among total distance.

a_i - demand at i-th node.

a_{max} , a_{min} - minimum and maximum among all demands.

The probability p_i of choosing node i as median calculated as below:

$$p_i = \frac{V_i}{\sum_{k=1}^{K} V_k} \quad , \ i = 1,2,....n; \qquad (3)$$

Where,

K - denotes the number of nodes that are not previously be chosen.

The partial solutions then will be evaluated based on the summation of total distance served by the median so far. After the evaluation, loyalty decision and recruiting process will be done using equation (4) and (5) under result and discussion chapter.

CHAPTER 3: METHODOLOGY OF STUDY

This chapter describes the research methods used for the research of BCO, TSP and P-Median problem. Research methodology is a set of procedures or methods used to conduct research. Documents review is the research method used to gain understanding of the basic concept of BCO and the problems. This research method is also important in this research paper to gather knowledge on how to apply BCO to the problems given. This chapter will also include the software development tools that will be used and also the gantt chart for the whole processes planned for this project.

3.1 RESEARCH METHODS

The research methods used for this dissertation purpose are the review of literatures, journals and books from the Internet and library. It is important to analyze and understand these papers in order to grab the fundamental of BCO and the problems.

3.1.1 Document Review Methods

Document reviews method is used in the research about BCO and the problems by reviewing the electronic and printed documents of journals and literatures from the internet. Those documents or reports are published based on the study of past and current research and findings. The findings or the results from this research method can be easily quantified and analyzed later.

3.1.2 Hands-On Methods

To gain the basic understanding of the concept of BCO required the author to use the hands-on method. It means that the author should play around with the coding itself. This practical approach could bring faster understanding of the concept suitable to be applied in this limited given time of research period.

3.1.3 Evolutionary Model

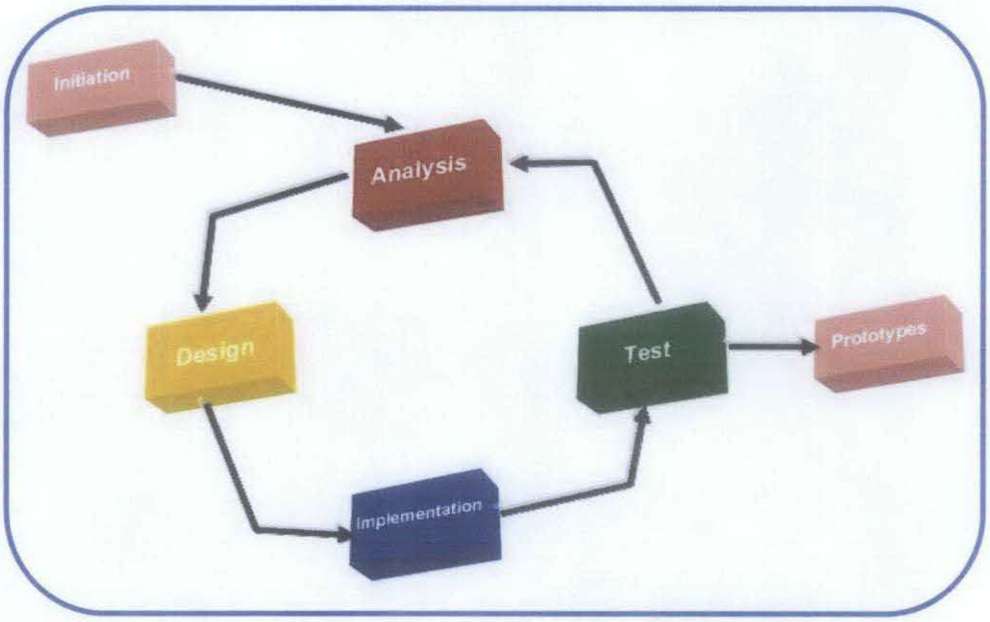


Figure 2: Evolutionary Model

Evolutionary model is a new model among the other models of system development life cycle. The model focus repetition of analysis, test, implementation and design to suits the needs of better understanding when dealing with programming. The best design will later be delivered as the prototype for future uses. This model suits this research the best as it allows flexibility of changing the coding and encourages detail research to be done before final output can be produced.

Initiation process is the starting point where thorough understanding of the BCO fundamental was being developed. It was done by a lot of reviews on related documents and researches that have been done previously on the equivalent project. The readings were really helpful to help in gathering the raw information about BCO fundamental. However, the fundamental got clearer when the process come to the main cycle consists of analysis, design, implementation and testing.

In analysis process, the understanding of mathematical equations and the overall process cycle of BCO was developed. It was the hardest time where new mathematical equation being involved in the process and it has to be understood before the project can really be started.

In this design section, the pseudocode to migrate BCO concept into a working prototype has been created. It was developed mainly to help the programmer in doing the coding in sequence and following exactly the BCO concept without a single error.

Implementation phase was the time where the coding was developed. In this research, the coding was developed into classes for reusability purposes. It also can be considered as one of the best practice in doing programming because the coding can be developed in more structured format and easier for understanding purposes.

Testing is where the partial solution of the project being evaluated. This is the crucial part to make sure that the program could deliver the best result based on the benchmark results given. The process of analysing information, designing the pseudocode, implementing the coding and the testing have been done repeatedly in a cycle to improve the coding or the prototype for producing better solution. For now, the prototype is still 30 percent behind in producing the optimum solution. So, the cycle is kept going up until now to find the problem and improve the prototype for future uses.

3.2 SYSTEM DEVELOPMENT TOOLS

3.2.1 Coding BCO Algorithm

Visual Studio will be used for designing and writing the codes for BCO solutions. C++ knowledge was required to maximize the use of Visual Studio in implementing the algorithm.

3.2.2 Comparing Results derived from TSP-ACS and TSP-BCO

Microsoft Excel will be used for statistical testing and analyzing results derived from solving TSP using BCO and ACS. The statistical data will be presented in graph, bar chart, table and other suitable comparison figures.

3.2.3 Importing standard data

Notepad will be used to store data sets and to be exported to Visual Studio during running time.

3.3 GANTT CHART

		Months									
		2010					2011				
No	**Activities**	**Aug**	**Sept**	**Oct**	**Nov**	**Dec**	**Jan**	**Feb**	**Mar**	**Apr**	**May**
1	**Planning**	■	■								
	Choose topic	🟦									
	Preliminary research on topic	🟦									
	Specify scope		🟦								
	Feasibility analysis		🟦								
2	**Analysis**		■								
	Understanding concepts		🟦								
3	**Design**			■	■						
	Design Algorithm Structure			🟦	🟦						
4	**Development**			■	■	■	■	■	■		
5	**Testing**						■	■	■		
	Comparison TSP-ACS and TSP-BCO						🟦	🟦	🟦		
6	**Implementation**								■	■	■

Figure 3: Project Gantt Chart

CHAPTER 4: RESULT and DISCUSSION

This chapter will discuss more details about the findings in Chapter 2. The discussion will focus on the BCO Metaheuristic algorithm and the equations, past results of BCO implementation on TSP and PMP problems.

4.1 BCO METAHEURISTIC ALGORITHM and EQUATIONS

BCO Metaheuristic algorithm follows almost the same way as the bees' nature of living. The algorithm is as follows:

1. Set prior algorithm parameter before the execution:
    - B –The number of bees in the hive
    - NC –The number of constructive moves during a single forward pass
2. At the start, all bees are in the hive. In simple word, during the *Initialization,* bees are assigned with no solution.
3. For each bee : // (the forward pass)

    (a) Set k =1; // (count constructive moves in the forward pass)

    (b) Evaluate all possible constructive moves

    (c) Choose one move using the roulette wheel:

    (d) $k = k + 1$;If $k \leq NC$ Go to step(b).
4. All bees are back to the hive ; // (backward pass starts)
5. Evaluate (partial) objective function value for each bee.
6. Each bee decides randomly whether to continue its own exploration and become a recruiter, or to become a follower.
7. For each follower, choose a new solution from recruiters by the roulette wheel.
8. If solutions are not completed Go to step 3.
9. Evaluate all solution and find the best one.
10. If the stopping criteria is not met Go to step 3.
11. Out put the best solution found.

Step 3 and step 4 are dependent on the problem (refer to 2.4 above) where the BCO Metaheuristic algorithm will be applied. For step 6 and 7, there formulas have to be applied for each step. The formulas are as below:

<u>For STEP 6: Loyalty Decision formula.</u>

$$p_b^{u+1} = e^{-\frac{O_{max} - O_b}{u}}, \quad b = 1, 2, \ldots, B, \qquad (4)$$

Where,

O_b – the normalized value for the objective function of partial solution created by the b-th bee;

O – maximum overall normalized values of partial solution to be compared;

u – the ordinary number of the forward pass (i.e., u=1 for the first forward pass, u =2 for second forward pass, etc.)

<u>For STEP 7: Recruiting Process formula.</u>

$$p_b = \frac{O_b}{\sum_{k=1}^{R} O_k}, \quad b = 1, 2, \ldots, R \qquad (5)$$

Where,

O_k - Represents normalized value for the objective function of the k-th advertised partial solution

R - Denotes the number of recruiters.

As mention above, there two alternating phases involved in BCO algorithm which are forward pass and backward pass. According to research done, the hive is a non-natural object with no precise location to be put in the search space and it also does not influence the algorithm execution. The hive is mainly used to denote the point where bees exchange information after every single forward pass before starting the backward pass alternately.

To further understanding of forward pass and backward pass, let us take an example of {Bee1, Bee2, …., Bee B] that engaged in solving a problem consisting of n number of components. Like mention above, we have to decide on how many constructive moves (NC) for the each bee to construct in a single forward pass. In this case, let us denote NC to be 1. It means that each bee should visit 1 node in a forward pass before it come back to hive for backward pass. The possible situation that might happen after 3rd forward pass illustrated as below:

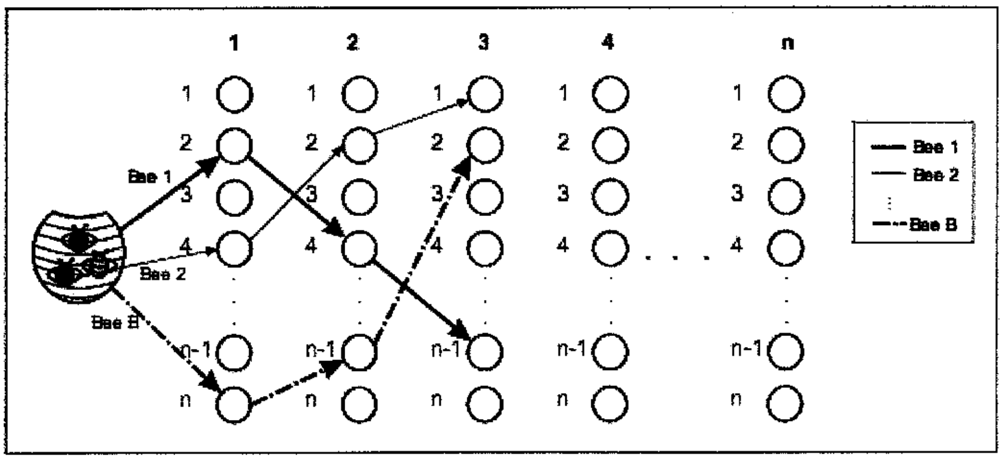


Figure 4: Illustration after 3rd forward pass

In the real process, after each forward pass or precisely after visiting 1 node in a single forward pass, bee will back to their hive and share on the quality of its partial solution. It then decide to loyal and keep visiting its current partial solution or following other bees' solution. This process is called backward pass and as shown in figure 4 above, we assume that every bee is loyal to their partial solution up until 3rd forward pass.

In the next forward pass, let assume that Bee 2 decided to be an uncommitted bee and choose to follow Bee B partial solution for 4th forward pass. When Bee 2 decides to follow Bee B in the next forward pass, it will copy the partial solution of Bee B up until 3rd forward pass before it continues its own path in the 4th forward pass. The illustration of that scenario is as below:

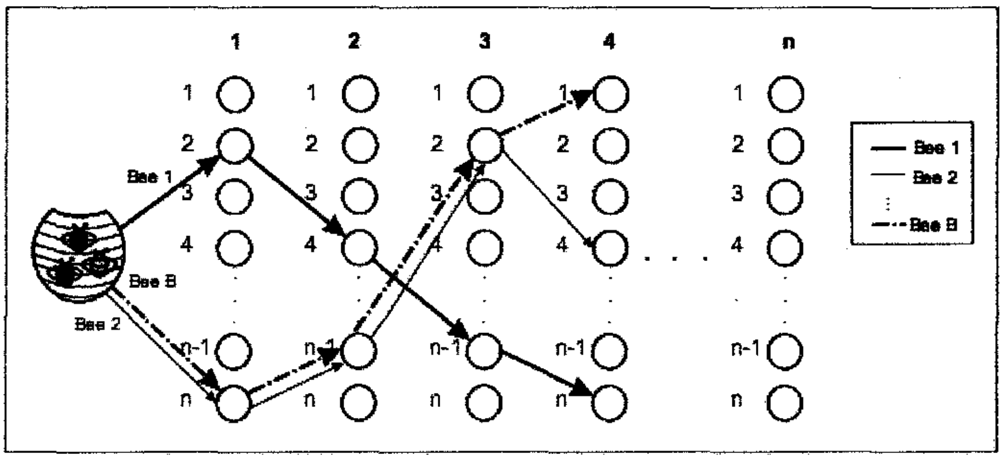


Figure 5: Illustration after 4th forward pass

All bees will alternately continue the process until they find all required feasible solution to update the global best solution. After updating the global best solution, all current solutions will be deleted to start a new iteration to find the optimum solution until the stopping criteria is met. The decisions in each forward and backward pass is connected to probabilities, loyalty decision and recruiting as per discussed earlier.

4.2 IMPLEMENTATION of BCO ALGORITHM

In this research, C++ has been used as the main programming tool to develop the coding for BCO algorithm. By using Visual Studio, BCO algorithm has been developed into classes for reusability purposes and it also one of the good programming approach in C++. The classes are as follow:

Class Node:

```
class Node{
private:
      int id;

public:
      int x,y;
      Node(){}

      Node(int id_,int x_,int y_){
            setId(id_);
            setXY(x_,y_);
      }

      void setId(int id_){
            id = id_;
      }

      void setXY(int x_,int y_){
            x = x_;
            y = y_;
      }

      int getId(){
            return id;
      }

      float getDistance(Node aNode){
            return float
            (sqrt(pow(x-aNode.x,2.0)+pow(y-aNode.y,2.0)));
      }
};
```

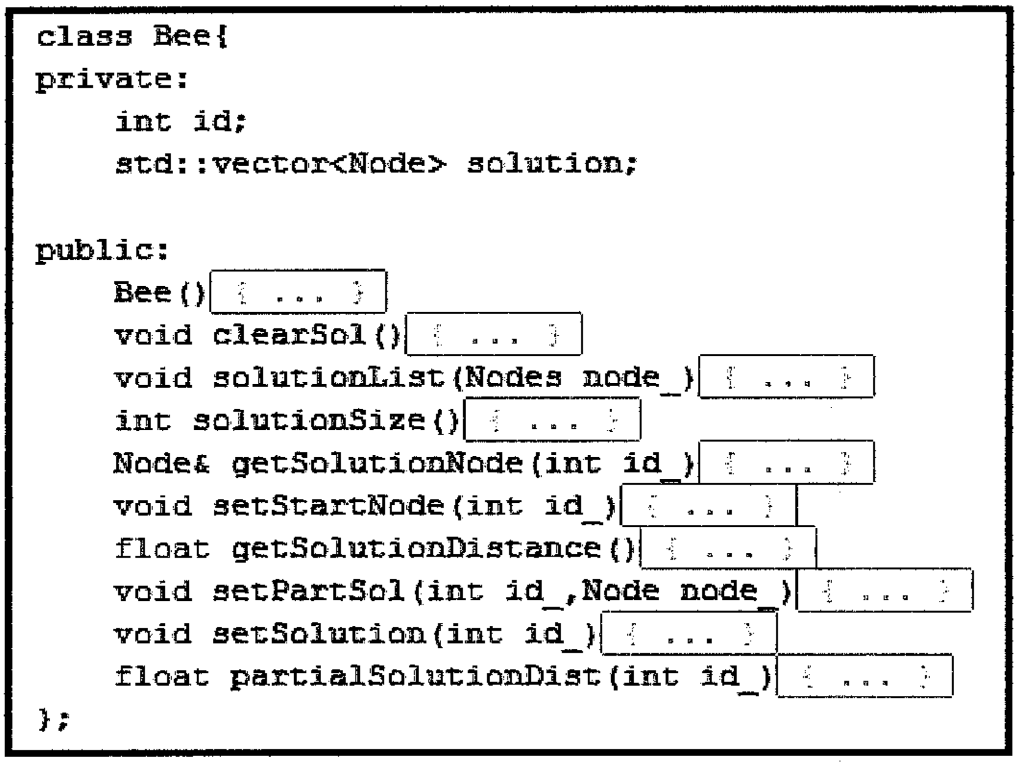
Basically, class node is used to store instances of TSP and PMP problem. The text file given consists of ID number for the node and the X and Y coordinate for the node. Using this class, we can reuse the datatype within the coding repeatedly. The function setId and setXY are used to allocate those instances to the program memory. Function getDistance in other hand is used to return distance between a pair of nodes.

Class Nodes:

```
class Nodes{
private:
      std::vector<Node> list;
public:
      Nodes(){}

      void addNode(Node node_){
            list.push_back(node_);
      }

      void getNodesFromFile(std::string filename_){
            FILE* input;
            input = fopen(filename_.c_str(),"r");

            if(!input){
                  printf("Error: File not found");
            }

            int num1 , num2 ,num3;
            while
            (fscanf(input,"%d %d %d",&num1,&num2,&num3)!=EOF)
            {
                  addNode(Node(num1,num2,num3));
            }

      }

      int nodesSize(){
            return list.size();
      }

      Node& getNode(int id_){
            return list[id_];
      }
};
```

Nodes class basically used to store an array of node. Instead of using array, vector was used to store the group of instances as the size of a vector is not fixed and can be changed accordingly within the coding. In this class, function getNodesFromFile was the main concern where all instances in the text file being scanned and stored in the program memory as an array. Function nodesSize is used to return the actual size of all instances and function getNode basically will return the whole array of node to the function call.

Class Bee:



```
class Bee{
private:
    int id;
    std::vector<Node> solution;

public:
    Bee() { ... }
    void clearSol() { ... }
    void solutionList(Nodes node_) { ... }
    int solutionSize() { ... }
    Node& getSolutionNode(int id_) { ... }
    void setStartNode(int id_) { ... }
    float getSolutionDistance() { ... }
    void setPartSol(int id_,Node node_) { ... }
    void setSolution(int id_) { ... }
    float partialSolutionDist(int id_) { ... }
};
```

This class is used to run the algorithm process on bee's individual basis. Basically, a bee will be assigned with unique id and will have its own set of solution. Below are the discussions on the use of each individual function in this class:

1) *clearSol function :*

   - clear the solution vector for individual bee at the beginning of each iteration.

2) *solutionList function:*

   - create a solution vector for each and every individual bee initialized in the beginning of the process.

3) *solutionSize function:*

   - return the size of the solution vector of each individual bee.

4) *getSolutionNode function:*

   - return the solution instances (node) to the function call.

5) *setStartNode function:*

   - randomize starting node at the beginning of the process.

6) *getSolutionDistance function:*

   - return the solution distance for solution found by each individual bee.

7) *setPartSol function:*

   - set the chosen pair of node into the solution vector for each bee as its partial solution during each forward and backward pass stage.

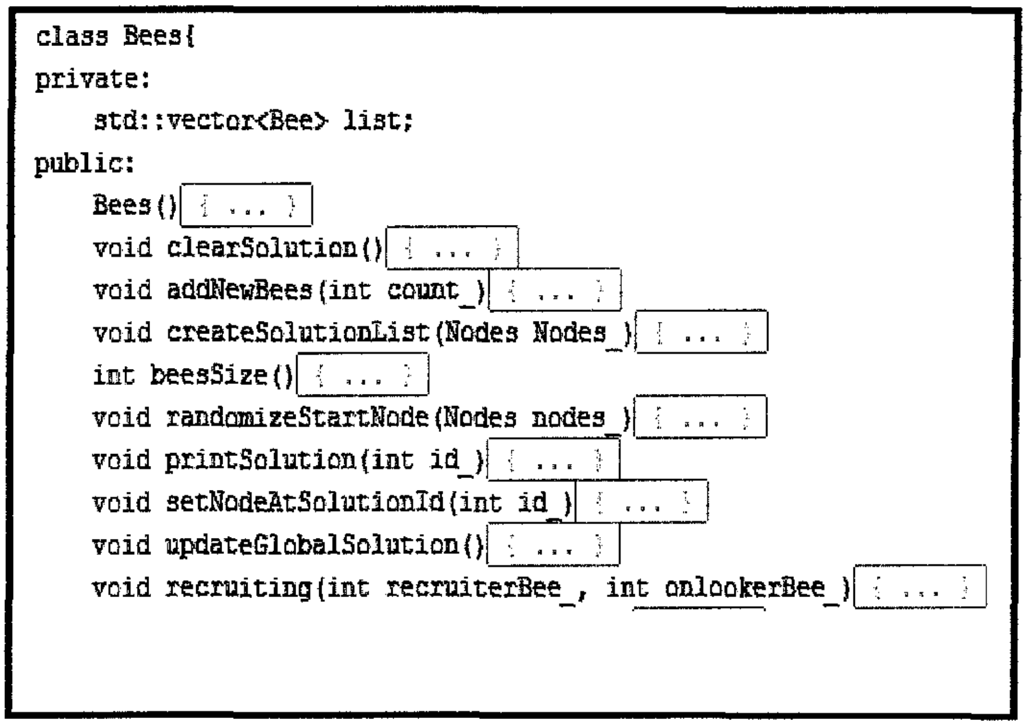
8) *setSolution function:*

   - this function is the path construction function where the pair wise of each node being choose as the solution.

9) *partialSolutionDistance function:*

   - return the distance for partial solution created so far by each individual bee.

Class Bees:



```
class Bees{
private:
    std::vector<Bee> list;
public:
    Bees(){ ... }
    void clearSolution(){ ... }
    void addNewBees(int count_){ ... }
    void createSolutionList(Nodes Nodes_){ ... }
    int beesSize(){ ... }
    void randomizeStartNode(Nodes nodes_){ ... }
    void printSolution(int id_){ ... }
    void setNodeAtSolutionId(int id_){ ... }
    void updateGlobalSolution(){ ... }
    void recruiting(int recruiterBee_, int onlookerBee_){ ... }
```

Class Bees is used to cater the process that involve decision making made by the group of bee. This can visualize the waggle dance process during the backward pass stage in the algorithm. The bee vector was created mainly to retrieve individual bee's solution or information in making the decision process. The use of each function will be discussed as below:

1) *clearSolution function:*

   - clear the solution vector for individual bee at the beginning of each iteration.

2) *addNewBees function:*

   - create the bee vector listing all the bee ids initialized at the beginning of the process.

3) *creatSolutionList function:*

   - create a solution vector for each and every individual bee initialized in the beginning of the process.

4) *beesSize function:*

   - return the size of bee vector to the function call.

5) *randomizeStartNode function:*

   - randomize starting node at the beginning of the process.

6) *printSolution function:*

   - print out the best solution found among the solutions gathered by all bees.

7) *setNodeAtSolutionId function:*

   - constructing the path in the bee's individual solution vector. Call function setSolution in Class Bee.

8) *updateGlobalSolution function:*

   - updating the best global solution that have been found so far by all bees.

9) *recruiting function:*

   - onlooker or uncommitted bees looking for partial solution advertised by recruiter bees. They will choose the best partial solution so far and copy the solution to their own solution memory.

10) *loyaltyDecision function:*

   - each bee decide whether or not to continue its own exploration. If not, they will go for recruiting process to follow their own choice of partial solution among the recruiter bees.

Main Function:

```
int main (){

      int forwPass,iter,noBee;

      Nodes nodes;
      Bees bees;
      srand(time(NULL));

      std::cout<<"Number of Iteration = ";
      std::cin>>iter;
      std::cout<<"Number of Bees = ";
      std::cin>>noBee;

      nodes.getNodesFromFile("tsp30.txt");

      forwPass = (nodes.nodesSize()-1);

      bees.addNewBees(noBee);

      for (int a=0;a<iter;a++){

            bees.clearSolution();

            bees.createSolutionList(nodes);

            bees.randomizeStartNode(nodes);

            for (int i=0;i<forwPass;i+=NC){
                  int temp=0;
                  forwardPass++;

                  for (int j=1;j<NC+1;j++){
                        bees.setNodeAtSolutionId(i+j);
                        temp=i+j;
                  }
                  bees.loyaltyDecision(temp,forwardPass);
            }

            bees.updateGlobalSolution();
            startPoint++;
            printf("Iteration = %d\n",a);
      }

      printf("\n\nBest Solution :\n");
      for (int i=0;i<bestGlobalSolutionNodes.size();i++) {
            int temp = bestGlobalSolutionNodes[i].getId();
            printf("%d\n",temp);
      }

//gl=======================================================gl//

      Graphic::mainGL(nodes);

      getch();
      return 0;
}
```

Displayed above is the main function for the algorithm. It basically consists of the overall process of initialization, forward pass, backward pass and the solution output. User will define the number of iteration and number of bees to use in the beginning of the process and the program will run until the condition is met.

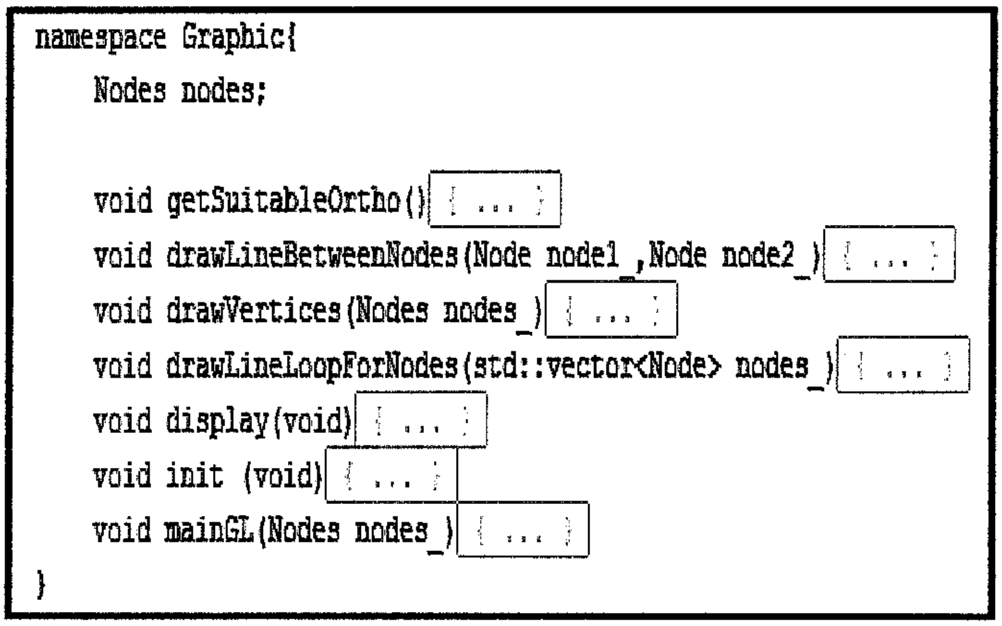
Besides those classes, the coding also involves the use of *namespace* to further sub-divide the code for simplicity matter. Namespaces allow the grouping of functions, classes, attributes and other into a single unique name for future use within the coding. Below are the namespaces that have been applied in the coding:

Namespace Fitness:

```
namespace Fitness{

      void creatFitness(Nodes nodes){
            for(int i=0;i<nodes.nodesSize();i++){
                  for(int j=0;j<nodes.nodesSize();j++){
                              fitness[i][j]=0;
                  }
            }
      }

};
```

Namespace fitness has been used to store weight of each individual pair of node using 2D matrix. This part has been discussed earlier (see 2.3) about the building block concept.

Namespace Graphic:



```
namespace Graphic{
    Nodes nodes;

    void getSuitableOrtho(){ ... }
    void drawLineBetweenNodes(Node node1_,Node node2_){ ... }
    void drawVertices(Nodes nodes_){ ... }
    void drawLineLoopForNodes(std::vector<Node> nodes_){ ... }
    void display(void){ ... }
    void init (void){ ... }
    void mainGL(Nodes nodes_){ ... }
}
```

Consists of several functions as below:

getSuitableOrtho function:

```
void getSuitableOrtho(){
      xbiggest = nodes.getNode (0).x ;
      xsmallest = nodes.getNode (0).x ;

      ybiggest = nodes.getNode (0).y;
      ysmallest = nodes.getNode (0).y ;

      for(int i=1; i<nodes.nodesSize(); i++)
      {
            if(nodes.getNode (i).x > xbiggest)
                  xbiggest = nodes.getNode (i).x;

            if(nodes.getNode (i).x < xsmallest)
                  xsmallest = nodes.getNode (i).x;

            if(nodes.getNode (i).y > ybiggest)
                  ybiggest = nodes.getNode (i).y;

            if(nodes.getNode (i).y < ysmallest)
                  ysmallest = nodes.getNode (i).y;
      }
}
```

This function used to put the suitable screen size during the graphic representation.

drawLineBetweenNodes function:

```
void drawLineBetweenNodes(Node node1_,Node node2_){
            glBegin(GL_LINES);
                  glVertex2f( node1_.x, node1_.y );
                  glVertex2f( node2_.x, node2_.y);
            glEnd();
}
```

This function is used to draw line between a pair wise of the nodes until it make a close loop which represent the solution that have been found by the bees.

drawVertices function:

```
void drawVertices(Nodes nodes_){
            glColor3f(0.9,0.1,0.1);
            glBegin(GL_QUADS);
                  for(int i=0;i<nodes_.nodesSize();i++){
                        glVertex2f( nodes_.getNode(i).x+0.8,
                        nodes_.getNode(i).y );

                        glVertex2f( nodes_.getNode(i).x,
                        nodes_.getNode(i).y+0.8 );

                        glVertex2f( nodes_.getNode(i).x-0.8,
                        nodes_.getNode(i).y );

                        glVertex2f( nodes_.getNode(i).x,
                        nodes_.getNode(i).y-0.8 );
                  }
            glEnd();
}
```

This function is used to draw the nodes point within the display area. The glColor3f sub-function defined the red colour used to draw the nodes point in the graphic presentation.

drawLineLoopForNodes function:

```
void drawLineLoopForNodes(std::vector<Node> nodes_){
      glColor3f(1.0,1.0,1.0);

      for(int i=1;i<nodes_.size();i++){
            drawLineBetweenNodes(nodes_[i-1],nodes_[i]);
      }
}
```

This function is used to draw the close loop of the solution found by the bees group.

display function:

```
void display(void){
      drawLineLoopForNodes(bestGlobalSolutionNodes);
      drawVertices(nodes);
      glutSwapBuffers();
}
```

This function is used to define the overall display output that will be presented in the display area.

init function:

```
void init (void) {
      /* select clearing color      */
      glClearColor (0.0, 0.0, 0.0, 0.0);

      /* initialize viewing values  */
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      glOrtho(xsmallest-5, xbiggest+5, ysmallest-5, ybiggest+5,
      -1.0, 1.0);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
}
```

This function is used to initialize the boundaries of the display area. The glClearColor is a sub-function that defines the background colour used in the display area.

mainGL function:

```
void mainGL(Nodes nodes_){
      nodes = nodes_;
      //glutInit(&argc, argv);
      glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
      glutInitWindowSize (400, 400);
      glutInitWindowPosition (100, 100);
      glutCreateWindow ("Result");
      //glutFullScreen();

      getSuitableOrtho();
      init();
      glutDisplayFunc(Graphic::display);

      /*glutKeyboardFunc(reset_tri);*/
      glutMainLoop();
}
```

This is the main function for the graphic presentation in the algorithm implementation.

4.3 PAST RESULT of BCO-TSP IMPLEMENTATION

Problem name	No. of nodes	Optimal value	BCO	Relative error (%)	CPU (sec)
Eil51	51	428.87	428.87	0.00	29
Berlin52	52	7544.37	7544.37	0.00	0
St70	70	677.11	677.11	0.00	7
Pr76	76	108159.00	108159.00	0.00	2
Kroa100	100	21285.40	21285.40	0.00	10
Eil101	101	640.21	640.21	0.00	61
Tsp225	225	3859.00	3899.90	1.06	11651
A280	280	2586.77	2608.33	0.83	6270
Pcb442	442	50783.55	51366.04	1.15	4384
Pr1002	1002	259066.60	267340.70	3.19	28101

Table 6: TSP benchmark problems: The results obtained by the BCO Metaheuristic algorithm taken from [13].

From the result above, BCO Metaheuristic proposed by Lucic and Teodorovic was proved capable of producing result nearest to the optimal value. In addition, the CPU processing times to acquire the results are low which conclude that BCO Metaheuristic was capable of producing good results in reasonable processing time.

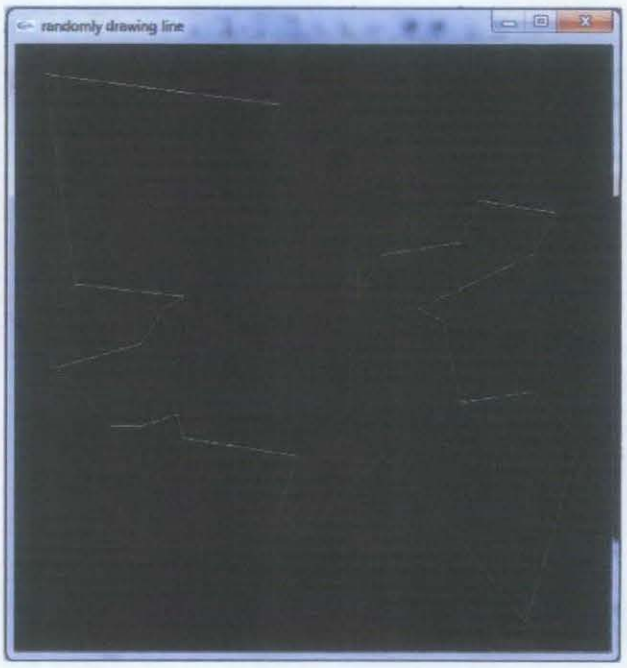


Figure 6: Benchmark Result BCO-TSP (Oliver30)

Figure 6 shows the benchmark result for TSP Oliver30 problem. It comprises of 30 nodes and the optimum solution for the problem is 423.74. In this research, it is targeted to at least meet the optimum solution using BCO. As far as the research goes, the best iteration to be able to find the optimum solution for now was done by ACO in only 30 iterations. BCO have to produce better result in obtaining the optimum solution and should come out with minimum processing time as possible.

4.4 EXPERIMENTAL RESULT BCO-TSP

Problem Name	No. of Nodes	Opt. Value	BCO Value
Wi29	29	27601.17	5308.87
Oliver30	30	423.74	614.29
Berlin52	52	7544.37	19768.10
St70	70	677.19	2510.97

Table 7: Experimental Result BCO-TSP

Result above obtained from testing done using prototype of BCO developed throughout the research. There are five problem being tested which include Wi29 (29 Nodes), Oliver30 (30 Nodes), Berlin52 (52 Nodes), St70 (70 Nodes) and KroA100 (100 Nodes). However, the prototype could not deliver the optimum result yet due to some missing limitation or miscalculation. The process of analysis, design, implementation and test phases are still in cycle to develop better prototype to produce the optimum solution within minimum processing time as possible.

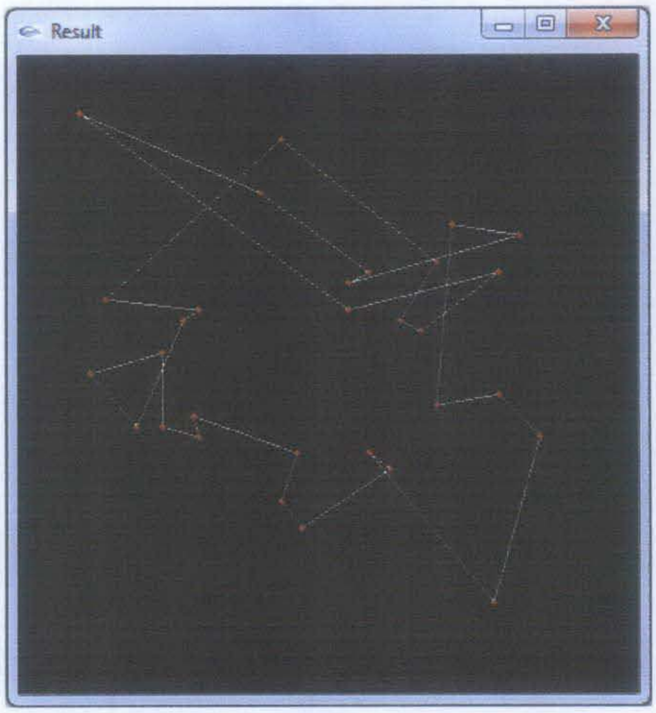


Figure 7: Experimental Result BCO-TSP (Oliver30)

Result above shows the graphic presentation of Oliver30 problem solved by the prototype. The figure shows that the result still way far behind the benchmark result and it is expected to obtain the benchmark result before the research ended.

4.5 PAST RESULT of BCO-PMP IMPLEMENTATION

Test problem	n	p	BCO Rel. error (%)	CPU (sec)	Test problem	n	p	BCO Rel. error (%)	CPU (sec)
M1	20	2	0	0.48	Set2	1000	2	0.60	112.00
M2	20	3	0	0.96	Set2	1000	4	4.45	448.00
M3	20	4	0	2.40	Set3	1000	2	0.34	95.40
M4	25	2	0	0.46	Set3	1000	4	2.94	211.40
M5	25	3	0	2.65	Set4	1000	2	0.55	117.00
M6	25	4	0	6.99	Set4	1000	4	3.02	206.00
M7	30	2	0	0.46	Set5	1000	2	0.26	117.00
M8	30	3	0	2.33	Set5	1000	4	4.34	406.00
M9	30	4	0	5.33	Set6	1000	3	1.14	168.00
M10	40	2	0	0.45	Set6	1000	5	7.48	448.00
M11	40	3	0	2.34	Set7	1000	3	1.09	102.10
M12	40	4	0	3.35	Set7	1000	5	4.40	300.30
M13	50	2	0	1.14					
M14	50	3	0.73	23.52					
M15	50	4	1.71	46.94					

Table 8: The p-median test problems: Results obtained by the BCO Metaheuristic algorithm taken from [13].

From table 2 above, it is also conclude that BCO Metaheuristic was best to be applied to the p-median problem. The algorithm seems able to produce nearest value to the optimal solution in a reasonable processing time. Below are more test data available for p-median problem on the internet:

Data File	Optimal Solution	Data File	Optimal Solution
pmed1	5819	pmed21	9138
pmed2	4093	pmed22	8579
pmed3	4250	pmed23	4619
pmed4	3034	pmed24	2961
pmed5	1355	pmed25	1828
pmed6	7824	pmed26	9917
pmed7	5631	pmed27	8307
pmed8	4445	pmed28	4498
pmed9	2734	pmed29	3033
pmed10	1255	pmed30	1989
pmed11	7696	pmed31	10086
pmed12	6634	pmed32	9297
pmed13	4374	pmed33	4700
pmed14	2968	pmed34	3013
pmed15	1729	pmed35	10400
pmed16	8162	pmed36	9934
pmed17	6999	pmed37	5057
pmed18	4809	pmed38	11060
pmed19	2845	pmed39	9423
pmed20	1789	pmed40	5128

***Data file available at:**

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/?C=M;O=A

CHAPTER 5: CONCLUSION and RECOMMENDATION

For now, the research has yet to find the evidences to prove that BCO is the best solution for those problems. However, based on the benchmark results from other researches, it seems that the idea of using BCO Metaheuristic to solve the TSP and PMP problem can be one of a good solution as the size of the solved problems could be increased by using the proposed algorithm. It is recommended to do a research on combining the two well known Metaheuristic algorithm, ACS and BCO in future research as it could lead to a better idea of solutions. Instead of just using one algorithm to solve the TSP or PMP, we combine the two algorithms which carry their own advantages in hopes of resulting in a more efficient result. BCO Metaheuristic has a problem in controlling parameters whereas ACS can be used in dynamic applications especially ones that requires adaptation to changes such as new distances. When combining both the algorithm, we can hope to see a much better result in finding the shortest tour for TSP and also better result in finding medians in PMP problem.

CHAPTER 6: REFERENCES

1) J.E.Beasley, August 2010

   Retrieved August 12, 2010, from <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>

2) Wikimedia Foundation, Inc. (August 4, 2010).

   Artificial Bee Colony Algorithm. Retrieved August 12, 2010, from <http://en.wikipedia.org/wiki/Artificial_Bee_Colony_Algorithm>

3) Wikimedia Foundation, Inc. (August 6, 2010).

   Bee Colony Optimization. Retrieved August 12, 2010, from <http://en.wikipedia.org/wiki/Bee_colony_optimization>

4) Artificial Bee Colony (ABC) Algorithm Homepage.

   Retrieved August 12, 2010, from <http://mf.erciyes.edu.tr/abc/index.htm>

5) Dr. Eugene M. Izhikevich. (July 26, 2010).

   Artificial Bee Colony Algorithm. Retrieved August 12, 2010, from <http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm>

6) McCafrrey J. D. (2009).

   Advance in Visual Computing: An empirical study of categorical dataset visualization using a simulated bee colony clustering algorithm. Retrieved August 12, 2010, from

   <http://books.google.com.my/books?id=2F_nPm9LjJEC&pg=PA182&lpg=PA182&dq=%22bee+system%22+algorithm&source=bl&ots=t9NJiKwwT2&sig=safM7vONwrTw2194Yn7H5oXKY8I&hl=en&ei=t6KzTLinEYXZcc-6hZ4I&sa=X&oi=book_result&ct=result&resnum=4&ved=0CCIQ6AEwAw#v=onepage&q=%22bee%20system%22%20algorithm&f=false>

7) M. Farooq. 2006.

   "From the Wisdom of the Hive to Intelligent Routing in Telecommunication Networks" PhD Thesis, Dortmund University.

8) Quijino N., Passino K. M. 2009.

   Honey Bee Social Foraging Algorithm for Resource Allocation: Theory and Application. Retrieved September 4, 2010, From <http://www2.ece.ohio-state.edu/~passino/PapersToPost/BeesIFDtempsubmitted.pdf>

9) Yang X. S. 2004.

Introduction to Computational Mathematics.

Retreived September 4, 2010, From <http://books.google.com.my/books?id=T_i0o_55MNwC&pg=PA224&lpg=PA224&dq=%22Virtual+Bee+Algorithm%22+Yang&source=bl&ots=d3-U7Sfr0f&sig=YUtfssBSjg-lugwRmZ1AI4khjR0&hl=en&ei=fBG1TIKLBMalcJO7lJYM&sa=X&oi=book_result&ct=result&resnum=1&ved=0CBQQ6AEwAA#v=onepage&q=%22Virtual%20Bee%20Algorithm%22%20Yang&f=false>

10) Karaboga D., & Akay B. (October 28, 2009).

A survey: algorithms simulating bee swarm intelligence. Retrieved from <http://www.springerlink.com/content/500886349544k37u/fulltext.pdf>

11) Wikimedia Foundation, Inc. (October 15, 2010).

Travelling salesman problem. Retrieved October 18, 2010, From <http://en.wikipedia.org/wiki/Generalized_travelling_salesman_problem>

12) Liu y., & Passino K. M. (30 March 2000).

Swarm Intelligence: Literature Overview. Retrieved October 18, 2010, From <http://www2.ece.ohio-state.edu/~passino/swarms.pdf>

13) Teodorovic D., Davidovic T., & Selmic M. (February 2010)

Bee Colony Optimization: The Applications Survey. Retrieved October 18, 2010, From < http://www.mi.sanu.ac.rs/~tanjad/BCO-ACM-Trans-Ver2.pdf>

14) Wong L.P., Low M.Y.H., & Chong C.S. (February 2010)

An Efficient Bee Colony Optimization Algorithm for Traveling Salesman Problem using Frequency-based Pruning. Retrieved April 5, 2011, From < http://www.mi.sanu.ac.rs/~tanjad/BCO-ACM-Trans-Ver2.pdf>