REACTIVE APPROACH FOR AUTOMATING EXPLORATION AND EXPLOITATION IN ANT COLONY OPTIMIZATION

RAFID SAGBAN

DOCTOR OF PHILOSOPHY
UNIVERSITI UTARA MALAYSIA
2016

Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to Universiti Utara Malaysia for any scholarly use which may be made of any material from my thesis.

Requests for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara Malaysia
06010 UUM Sintok

Abstrak

Pengoptimuman koloni semut (ACO) boleh digunakan untuk menyelesaikan masalah sukar polinomial tidak berketentuan. Penerokaan dan eksploitasi adalah mekanisme utama bagi mengawal carian dalam ACO. Carian reaktif adalah satu teknik alternatif untuk mengekalkan kedinamikan mekanisme ini. Walau bagaimanapun, teknik carian reaktif berasaskan ACO mempunyai tiga (3) masalah. Pertama, model memori yang merakam kawasan carian yang terdahulu telah tidak memindahkan struktur sekitaran secara lengkap kepada leleran berikutnya yang akan membawa kepada permulaan semula dengan sewenang-wenangnya dan carian setempat yang pramatang. Kedua, penunjuk penerokaan adalah tidak teguh disebabkan oleh perbezaan magnitud dalam matriks jarak bagi populasi semasa. Ketiga, teknik kawalan parameter yang menggunakan penunjuk penerokaan dalam proses maklum balas telah tidak mempertimbangkan masalah keteguhan penunjuk. Satu algoritma pengoptimuman koloni semut reaktif (RACO) telah dicadangkan untul mengatasi kekurangan carian reaktif. RACO terdiri daripada tiga komponen utama. Komponen pertama adalah satu algoritma *max-min ant system* reaktif untuk merakamkan struktur sekitaran. Komponen kedua adalah satu mekanisme pembelajaran mesin berstatistik yang dinamakan *ACO*ustic untuk menghasikan penerokaan yang teguh. Komponen ketiga adalah algoritma pemilihan parameter mudah suai berasaskan ACO untuk menyelesaikan masalah pemparameteran yang bergantung kepada kualiti, penerokaan dan kriteria berpadu untuk memberi ganjaran kepada parameter yang berpotensi. Prestasi RACO dinilai menggunakan masalah jurujual kembara dan umpukan kuadratik dan dibandingkan dengan lapan (8) teknik metaheuristik berdasarkan kadar kejayaan, pangkat tanda Wilcoxon, Chi-square dan relatif peratusan sisihan. Hasil kajian menunjukkan prestasi RACO adalah lebih baik dari lapan (8) teknik metahuristik dan ini mengabsah keberkesanan RACO boleh digunakan sebagai satu hala baru bagi penyelesaian masalah pengoptimuman. RACO boleh digunakan untuk menyediakan mekanisme penerokaan dan eksploitasi yang dinamik, menetapkan nilai parameter yang membolehkan carian yang cekap, menerangkan jumlah penerokaan yang dilaksanakan oleh algoritma ACO, dan mengesan keadaan genangan.

Kata kunci: Pengoptimunan koloni semut, Carian reaktif, Dinamik penerokaan dan eksploitasi, Polinomial tidak berketentuan, *Max-min ant system.*

Abstract

Ant colony optimization (ACO) algorithms can be used to solve nondeterministic polynomial hard problems. Exploration and exploitation are the main mechanisms in controlling search within the ACO. Reactive search is an alternative technique to maintain the dynamism of the mechanics. However, ACO-based reactive search technique has three (3) problems. First, the memory model to record previous search regions did not completely transfer the neighborhood structures to the next iteration which leads to arbitrary restart and premature local search. Secondly, the exploration indicator is not robust due to the difference of magnitudes in distance matrices for the current population. Thirdly, the parameter control techniques that utilize exploration indicators in their feedback process do not consider the problem of indicator robustness. A reactive ant colony optimization (RACO) algorithm has been proposed to overcome the limitations of the reactive search. RACO consists of three main components. The first component is a reactive max-min ant system algorithm for recording the neighborhood structures. The second component is a statistical machine learning mechanism named *ACO*ustic to produce a robust exploration indicator. The third component is the ACO-based adaptive parameter selection algorithm to solve the parameterization problem which relies on quality, exploration and unified criteria in assigning rewards to promising parameters. The performance of RACO is evaluated on traveling salesman and quadratic assignment problems and compared with eight metaheuristics techniques in terms of success rate, Wilcoxon signed-rank, Chi-square and relative percentage deviation. Experimental results showed that the performance of RACO is superior than the eight (8) metaheuristics techniques which confirmed that RACO can be used as a new direction for solving optimization problems. RACO can be used in providing a dynamic exploration and exploitation mechanism, setting a parameter value which allows an efficient search, describing the amount of exploration an ACO algorithm performs and detecting stagnation situations.

Keywords: Ant colony optimization, Reactive search, Dynamic exploration and exploitation, Nondeterministic polynomial, Max-Min ant system.

Acknowledgement

In the Name off Allah, Most Gracious and Most Merciful

All praise and thanks go to the Almighty Allah SWT for giving me the guidance, good health and the strength to pursue my PhD. It is only with His Blessings finally this Ph.D journey could be finished. *Alhamdulillah*!

At the beginning of the Ph.D journey, I was always thinking about its end, but paradoxically I am now thinking about its beginning! My Ph.D has definitely its usual ups and downs, but thank you Prof. Dr. Ku Ruhana Ku Mahamud, for the continuous support, insight and patience during this long journey. Thank you for encouraging my research and for allowing me to grow as a research scientist. You have been a tremendous mentor and great supervisor. Your advice on both research as well as on my career have been priceless. I would also like to thank my co-supervisor for his brilliant comments and suggestions. Thanks to you Dr. Muhamad Shahbani Abu Bakar for being there to support me. Prof. Dr. Ku Ruhana and Dr. Muhamad Shahbani, without your constant trust and, sometimes, gentle prodding, this manuscript would not have been completed. Besides my supervisors, I would like to thank the thesis committee members, Prof. Dr. Rosni Abdullah, Prof. Dr. Suhaidi Hasan and Dr. Yuhanis Yusof, for every moment they have spent for reading my thesis and providing useful feedback.

To my parents, I hope you are proud to have me as your son. Mum and dad, I believe your consistent prayers, encouragement, and support over the years have made all of this possible. Needless to say, I'm indebted to my beloved wife, Ikhlas, for leaving Iraq to follow and to support me into my personal dream of getting a Ph.D in Swarm Intelligence. Thank you, with all my heart, for everything. Your unconditional love was what sustained me thus far. To my lovely children, Maryam, Zahraa, Hasanein and Rand, thank you baba for your love, understanding and patience. It goes without say, words cannot express how grateful I am to my brothers, Nahudh, Muqdad, Mushriq, Mohammed and Ali, for all of the sacrifices that you've made on my behalf.

I am deeply thankful to the management of University of Babylon for the study leave granted to me to finalize my Ph.D. Next, I am thankful for all units of Universiti Utara Malaysia for their kind help and endless support. Thanks to all collagenous in Internetworks Research Lab and IEEE UUM Student Branch who have become good friends for all scientific input, in the form of presentations and discussions. Over the years, this group has included; Dr. Adib M. Monzer Habbal, , Dr. Naseer Ali, Mohammed Alsamman, Abdullahi Ibrahim, Shivaleela Arlimatti, Sushank Chaudhary, Swetha Goudar, Ikram Ud Din and Walid Elbreiki. I will definitely miss all of you. Heartfelt thanks to Prof. Dr. Saad Talib, Prof. Dr. Wesam Bhaya, Dr. Mustafa Muwafak, Dr. Shafinah Farvin bt. Packeer Mohammed, Dr. Hanaa Kadum, Dr. Ashwak Alabaichi, Dr. Ahmed Talib and Dr. Ghassan Nashat who have incited me to strive towards my goals.

To my friends, Atheer Flayh, Raaid Alubady, Haydar Abdulameer Marhoon, Haydar Gubashi, Mehdi Ebadi, Emad Al-Anbari, Wissam Abdul Adheem, Salam Ghanim, Salih Glood, Zaid Khudir, Azhar Witwit and Bayadir Abbas, Hussain Mehdi, Abo Obaida Al-Sa'doon, Ali Ibrahim and Shaimaa Ali, Ahmed Gazi, Moheimen Mohomed, Ahmed Sheet, Khalid Al-Khafaji, Atyaf Sami, Samara Raheem, Emad Qasim, Alaa Ahmed and many more, thanks to all of you for making me feel like home and bring Iraq to me here. You all made my stay in Malaysia nicer and funnier, thank you all, hope we will stay in touch. The moments with you will not be forgotten ever.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
*ACO*ustic	Acoustic Mimicry Based Exploration Indicator
ACS	Ant Colony System
AFSA	Artificial Fish Swarm Algorithm
AI	Artificial Intelligence
ANISM	Average Number of Iterations to a Solution Measure
ANTabu	Ant Colony Based Tabu Search Algorithm
AntNet	Ant-based network routing algorithm
Ant-Q	Ant-Q-Learning
ANTS	Approximate Nondeterministic Tree Search
APC	Adaptive Parameter Control
APS_{ACO}	ACO-based Adaptive Parameter Selection
AS	Ant System
RAS	Ranked Ant System
ATSP	Asymmetric Traveling Salesman
Beam-ACO	Beam search based ACO
BWAS	Best-Worst Ant System
CbM	Components-based Memory
CG	Construction Graph
CO	Combinatorial Optimization
CTM	CPU Time Measure
DPL	Differential Path Length
E&E	Exploration and Exploitation
EAS	Elitist Ant System
EC	Evolutionary Computation
EP	Evolutionary Programming
GA	Genetic Algorithm
GH	Genetic Hybrid
HAFSO	Hybrid Artificial Fish-School Optimization
HAS	Hybrid Ant System
HCF	Hyper-Cube Framework
LB	Lower Bounds
MACO	Multiple Ant Colony Optimization
MMAS	Max-Min Ant System
MWVC	Minimum Weight Vertex Cover
MWVCP	Minimum Weight Vertex Cover Problem
NOG	Nonobjective Function Guided
NP-hard	Nondeterministic polynomial hard problems
OG	Objective Function Guided
OG-ACO	Object-Guided Ant Colony Optimization
PACO	Population Based ACO Approach

PbM	Population-based Memory
PSO	Particle Swarm Optimization
PTS	Pheromone Trail Smoothing
QAP	Quadratic Assignment Problem
QSM	Quality Solution Measure
RACO	Reactive Ant Colony Optimization
RH's	Reactive Heuristics
RL	Reinforcement Learning
RLS	Recursive Local Search
RMMAS	Reactive Max-Min Ant System
Ro-TS	Robust Tabu Search
RPD	Relative Percentage Deviation
RTS	Reactive Tabu Search
SA	Simulated Annealing
S-ACO	Simple-Ant Colony Optimization Algorithm
SI	Swarm Intelligence
SRM	Success Rate Measure
TS	Tabu Search
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem

CHAPTER ONE
INTRODUCTION

In the field of Artificial Intelligence (AI), significant numbers of mathematical problems are of practical and theoretical importance. An example of these problems is the *combinatorial problem* where we try to find the values for discrete variables. This is done particularly to satisfy certain specific conditions. The combinatorial problem can be further categorized into *optimization* and *satisfaction* problems.

The optimization problem is aimed to find an optimal set of discrete objects. This set is known as the *optimal solution* of other *candidate solutions (i.e., solution space)*. The objects of each of these solutions are called *solution components* (Bertsimas, Brown, & Caramanis 2011; Fletcher, 1997). On the other hand, the satisfaction problem is aimed to find a solution whose state satisfies a number of constraints or limitations. A *problem* is a scenario needed to be solved. An *instance* is a specific case of that scenario. For many combinatorial problems, the solution space for a given instance is large. As a result, it is not possible to be searched because of the functional dependency required between the size of any instance and the time and space to solve it. This is defined as the *complexity* of the problem (Carterette, 2011; Garey & Johnson, 1979).

In the complexity theory, there are two classes of problems: *polynomial* and *nondeterministic polynomial* (Korte & Vygen, 2006). A problem that is as hard as any problem in the non-polynomial class is called *NP-hard.* There is no *exact* algorithm that can be used to find an optimal solution for NP-hard problems in

polynomial time. This objective has been replaced with finding good solutions in a reasonable time by the use of *heuristic algorithms*. This class of algorithms can be classified into *constructive* or *iterative* methods. This classification is done according to its way of generating solution. Heuristic algorithms can be further classified into *stochastic* or *deterministic* methods, based to its way of search.

A constructive method always builds solution components from an empty set until it finds one candidate solution. An iterative method takes an arbitrary solution as an initial solution and modifies it iteratively. Both constructive and iterative methods employ the stochastic and deterministic methods. A stochastic method utilizes randomization to traverse the search space, while a deterministic method does not use randomization in its function. Moreover, deterministic search repeats its procedure each time it is applied to the same problem instance, but stochastic search may perform it differently (Hoos & Stützle, 2005; Rothlauf, 2011). However, the heuristic methods are restricted by the environment of the problem at hand. And this basically allows the search to be trapped in local optima, which is not the global optimal solution in search space of the CO problem under tackle.

To address the problem of local optima, new search methods have emerged which allow robust diversification to be performed in the search space. These methods are called *metaheuristics*. They basically combine heuristic methods in higher-level metaphors. Examples of these metaphors are annealing, memory, evolution, and ant foraging behavior. The metaheuristics that are inspired from the stated metaphors are: simulated annealing (SA), tabu search (TS), evolutionary computation (EC), and ant colony optimization (ACO) respectively (Gendreau & Potvin, 2010).

A metaheuristic method is defined as "an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions" (Osman & Laporte, 1996; Zufferey, 2012). Metaheuristics are divided into *local search* and *population-based* techniques. The local search technique manipulates single solution by exchanging segments of its components to produce better solutions while the population-based technique uses more than one solution. SA and TS algorithms belong to the former class, while GA and ACO belong to the later one. The search behavior differs from one metaheuristic to another based on the metaphor that the specific algorithmic components belong to. These components are *exploration and exploitation (E&E)* components as shown in Table 1.1.

Table 1.1

The Basic E&E Components in Metaheuristics

Metaheuristic	**E&E component**
1.SA	Acceptance criterion + cooling schedule
2.TS	Neighbor choice (tabu lists) aspiration criterion
3.EC	Recombination + mutation + selection
4.ACO	Pheromone update + probabilistic construction

Exploration refers to the probing of unvisited regions within the search space, while *exploitation* refers to the search around good solutions in the current problem space regions. The dynamic balance between exploration and exploitation is essential in order to find new regions quickly and to reduce the search time in regions that have already been explored (Beer, Hendtlass, & Montgomery, 2012).

In other words, any metaheuristic algorithm should be designed in a way that considers the E&E balance role in its search behavior. The single-solution based metaheuristics are more exploitation-oriented, whereas the basic population-based metaheuristics are more exploration-oriented (Boussaïd, Lepagnot, & Siarry, 2013).

In this light, ACO is grown as a population-based, metaheuristic, stochastic and constructive method for solving Combinatorial Optimization (CO) problems (Baghel, Agrawal, & Silakari, 2012). ACO is a popular framework in *Swarm Intelligence (SI)* (Merkle & Middendorf, 2005). SI utilizes the collective behavior of social insects to design algorithms or distributed problem-solving devices. In SI, swarms (i.e., agents) adapt quickly to the problem's environment without reprogramming. This flexibility and robustness motivate several successful applications of ant algorithms (Mohan & Baskaran, 2012).

ACO is inspired by the food foraging behavior of real ants. Once an ant finds a food source, the ant returns to its nest, depositing a chemical substance called *pheromone* to and from the nest. This trail will now guide other workers from the nest to the food source. The other ants return to the nest and deposit their own pheromone along the trail to reinforce their path. Therefore, the trail construction is a result of a *positive feedback* mechanism, the main component of the self-organization in the social insects (Bonabeau, Dorigo, & Theraulaz, 1999). It expresses that: the more these ants use a trail, the more attractive the trail becomes. A large number of foragers will quickly assemble around a food source. This cooperation also enables a colony to find the shortest path leading to a food source, and if the reinforcement

becomes too low after some time, the trail will disappear (Martens, Baesens, & Fawcett, 2011).

The second ingredient of the self-organization in social insects is the *negative feedback* which uses to counterbalance the positive feedback: it may take the form of food exhaustion or saturation (Garnier, Gautrais, & Theraulaz, 2007). The type of indirect communication between ants is known as *Stigmergy*. This was originally proposed by Grasse in 1950s (Bonabeau et al., 1999). Grasse opined that the building activity depends on the colony and not on the workers ants themselves (Theraulaz & Bonabeau, 1999). Notably, there were some experiments performed with real ants. These experiments showed how the Stigmergy mechanism can find the shortest path between the ant nest and the food source (Deneubourg, Aron, Goss, & Pasteels, 1990). This mechanism played a main role in designing the first ACO algorithm, namely ant system (AS), which is the base of subsequent ACO algorithmic frameworks (Dorigo, Gambardella, Middendorf, & Stützle, 2002).

In ACO, a colony is a set of *artificial ants* which cooperates to find the best solution to a CO problem. These ants generally modify a sequence of numeric values associated with different states of the problem. This sequence is known as the *artificial pheromone trail*. The pheromone trail is the sole means of communication among artificial ants (Dorigo & Socha, 2007). Several ACO extensions have been proposed to solve new combinatorial optimization problems and to reach a balance point between exploration and exploitation as well. The max-min ant system (MMAS) algorithm is a prominent extension of ACO framework which presents

high quality solutions, together with the proof of convergence to the optimal solutions (Dorigo & Stützle, 2004).

Generally, the exploration and exploitation balance is achieved by the proper management of a probabilistic memory model, i.e. the pheromone trail. This can be achieved using two complementary processes: probabilistic solution construction and pheromone update (Blum & Roli, 2003). An optional process of local search might be inserted to improve the quality solutions produced by probabilistic solution construction. In this way, the search concentrated quickly around high quality regions of search space which lead to premature convergence especially with large search spaces. One of the generic strategies to avoid the premature convergence is restarting the search based on some exploration triggers.

Other exploration and exploitation components are the strategic parameters to be adjusted by the designer or the practitioner of the algorithm, by the algorithm itself, or by other adaptation algorithms (Dorigo, Maniezzo, & Colorni, 1991; Lopez-Ibanez, 2010). These components are recruited to avoid convergence because of the sensitivity of ACO search to the parameters' selection. Reactive search is a framework (Battiti, Brunato, & Mascia, 2008) that integrates machine learning techniques with local searches together with online parameters' selection and restarting the search when the premature convergence occurs. An exploration indicator is harnessed as a trigger for restarting the search and as evidence for parameters adaptation. Overall, the ability of reactive search as a new technique to maintain the dynamism of the exploration and exploitation mechanics entails

integrating it with ant colony optimization to produce powerful approach for nondeterministic problem solving.

1.1 Problem Statement

The exploration versus exploitation dilemma arises when promising regions of search space need to be quickly identified without spending too much time in poor regions (Talbi, 2009). MMAS, the prominent ACO variant, has a relatively long initial exploration to avoid the quick convergence toward local optimum, where the algorithm is not able to generate new global solutions as run time passes (Maur, Stützle, & López-Ibáñez, 2010). Subsequently, the current memory model that records previous search regions is not able to completely transfer the neighborhood structures of current iteration to the next iterations which leads to an arbitrary restart and premature local search. Reactive search is a technique for automating exploration and exploitation using memory features and machine learning approaches for exploration indication (Battiti et al., 2008). The exploration indication (Pellegrini & Favaretto, 2012) in ACO-based reactive search is suffering a problem of robustness: Different circumstances entail assigning new value to the neighborhood threshold. The instability in threshold value assignment gets worse as fitness landscape flatten or local search procedure changed. Moreover, the performance of parameter adaptation methods is worsen if the standard CO problems are used or if more parameters are adapted (Pellegrini, Stützle, & Birattari, 2012). This is the result of involving poor indication schemes to evaluate the effect of the adaptive parameters' selection on the search. Therefore, solving problems in the combination of successful local search and restarting procedures with the aid of

advanced memory features, together with adaptive parameters' selection procedures and robust indication is an approach for effective ACO-based reactive search in order to automate the exploration and exploitation balance.

1.2 Research Questions

This research tries to answer the following questions:

- How does the scheme of memory improve the restart and local search mechanisms in MMAS?
- How can the exploration indicators perform more robustly in ACO algorithm?
- How can a robust exploration indicator contribute in online parameter selection?
- Does the combination of those reactive procedures improve the exploration and exploitation balance within the ACO algorithm?

1.3 Research Objectives

The main objective of this study is to propose a reactive approach for automating exploration and exploitation in ACO. The specific objectives are:

- To develop a memory model for improving restart and local search mechanisms.
- To enhance the exploration measurement in ACO in terms of robustness of indication.
- To propose an adaptive parameters' selection method based on robust indication.
- To evaluate the performance of the proposed approach.

1.4 Significance of the Research

The exploration and exploitation balance is crucial for better ACO performance. Reactive search is a technique for automating that balance by integrating machine learning and optimization in an online manner ("learning while optimizing"). The proposed approach improves the abilities of ACO in problem solving and addresses the circumstances that emerged from the ACO-based reactive search integration by:

- Defining a new memory model for ACO.
- Developing more effective ACO variants.
- Improving the exploration indication in ACO.
- Solving the parameterization problem by the intelligent tuning of parameters.
- Providing a well-balanced exploration and exploitation mechanism for ACO method.

The proposed approach can be applied for real-world applications when domain-specific knowledge is available. The applications include industry applications such as industrial vehicle routing, car sequencing, power distribution applications such as voltage control and electric power distribution, telecommunications applications such as traffic grooming in optical networks and biological applications such as bioinformatics. All of the applications require optimal solutions that would benefit from the balance between exploration and exploitation.

1.5 Scope of the Research

This research proposes a new algorithmic approach for controlling the exploration/exploitation behavior in the standard ACO that is designed to solve single-objective, static, combinatorial optimization problems. To achieve such goal,

the optimization problem of traveling salesman problem (TSP) and the quadratic assignment problem (QAP) have been chosen as test-beds for the experiments of this research. There are several ant algorithms not fitting into the standard ACO metaheuristic framework, e.g. Fast Ant System algorithm. This research concentrates on the standard approach and omitted other approaches because they differ from ACO algorithms mainly in some aspects.

For optimization problems to be solved by ACO, they have to be encoded in several ways. In this research, the feasible solutions are encoded using the concept of construction graphs. For most problems, there are alternative ways of encoding these solutions. Finding out which one is the best is out of the scope of this research. The solutions are constructed by the walk of ants through the *construction graph*. The dynamic problem-solving, i.e. changing the graph of the problem during the run, is also not considered.

The parameters' selection in ACO is problem independent. One of the disciplines for solving the problem is by following multiobjective optimization through operating more than objective functions: one for the problem under solving and one for the parameters' selection problem. This approach is limited to the single-objective handling. However, it is easy to extend it to a multiobjective function by separating the objective function for the second problem from the one of the problems to be solved. The proper parameter values that are involved in the search process are changed during the run. Parameter tuning is not part of the scope of this research because it is trying to address the problem in an offline way, i.e. before the run.

1.6 Thesis Organization

The remainder of this thesis is organized as follows. **Chapter Two** defines the concept of combinatorial optimization and the methods used to solve it. The chapter also outlines the biological inspiration and problem representation of ACO metaheuristic together with the prominent ACO algorithms as exemplified by MMAS. The rest of the chapter is divided into three divisions which focus on memory-based E&E strategies, exploration indication strategies and reactive-based parameters' selection strategies. **Chapter Three** addresses the experimental methodology used in implementing this research. After presenting high level abstraction of research framework, the summaries of three proposed memory model development, exploration indication enhancement and the proposal of adaptive parameters' selection are provided. Descriptions on TSP and QAP are presented followed by explanation of benchmark methods and comparative measures. **Chapter Four** introduces the development of memory model based on the component-based and population-based schemes. The chapter starts by identifying the optimal point to start through experimental analysis for several ACO models. In addition, this Chapter introduces reactive heuristics and recursive local search technique based on component-based and population-based memory schemes respectively. **Chapter Five** presents twofold of E&E components: the exploration measurement and the adaptive parameters' selection in ACO. For the first method, it describes the definition, modelling and implementation of a nature-inspired exploration indicator called *ACO*ustic. It was done by combining clustering information with statistical information gathered during the run. It has been analyzed and compared to the state-of-the art indicators in ACO literature. For the second component, this chapter

presents the development and evaluation of three variants of ACO-based adaptive parameters' selection algorithm by which the parameters' selection problem in ACO is addressed. The performance of reactive ant colony optimization approach, namely RACO is described in **Chapter Six**. The evaluation was done based on the experimental comparison approach to look into the impact of combining the proposed exploration and exploitation components in the above mentioned chapters. **Chapter Seven** concludes and outlines future directions of research. The chapter recalls the developed algorithmic components in the thesis and highlights the contributions made throughout the research. Finally, the thesis ends with suggestions for future works if any researchers wanted to embark on this kind of research.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

This chapter provides the background of the problem of this research and surveyed what have been done to solve the problems of E&E in ACO. A description about CO problems and their solving approaches; a more in-depth description about ACO and its various aspects; and the main E&E strategies in MMAS, are highlighted are in Sections 2.2 - 2.4. The background and perspective related works about each of the three E&E aspects are provided in this chapter as follows. Firstly, the memory-based E&E strategies in ACO are discussed in Section 2.5. Secondly, the exploration measurement tools needed for controlling E&E are presented in Section 2.6. Thirdly, the adaptive parameter's selection approaches in ACO are presented in Section 2.7. Section 2.8 provides a unified review about the abovementioned related works, i.e. the memory-based models, the exploration measures and parameter adaptation methods, while the chapter is summarized in Section 2.9.

2.2 Combinatorial Optimization Problems

Combinatorial (i.e. discrete) problems in AI can be classified as either optimization or satisfaction problems (Bertsimas et al., 2011; Fletcher, 1997). This research is focused on combinatorial problems in terms of optimization. To concentrate on optimization problems and not satisfaction problems is not a limitation, because any satisfaction problem can be formulated as an optimization problem. A CO problem is either a maximization problem or a minimization problem with an associated set of

instances (Korte & Vygen, 2006). This research focuses on minimization problems, as a maximization problem can easily be converted into a minimization problem. Each instance of CO problems can be represented as a tuple (S, f, Ω), where S is a set of candidate solutions and f is the objective function which is assigned to every s ∈ S a value of f (s). The goal of optimization then is to find a solution (s) with a minimal f(s) (i.e. minimal cost). This solution called the globally optimal solution to the problem (S, f, Ω), and denoted by f (s_{opt}). Ω is a set of constraints (e.g. in TSP route, each city has to be visited exactly once and that route has to start and end at the same city. Finding that route with a minimal cost is the task of artificial ants during the optimization process. This route is the globally optimal solution to the problem of TSP which is denoted by f (s_{opt})). The way to solve (S, f, Ω) problems is by enumerating *all* set of solutions (S) and picking the one with minimal cost. Following the complexity of the problem, it is infeasible for many problems as the size of the search space, denoted by |S|, grows exponentially with instance size (Carterette, 2011; Garey & Johnson, 1979). Hence, the kind of trade-off between the quality of solution and the computational efforts has to be considered. Several optimization methods are proposed in order to find the (near-) optimal solution to the problems based on the mentioned considerations.

The optimization of the problem solves exactly or approximately based on the complexity of the problem as in Figure 2.1. The exact methods guaranteed the optimality of their solutions. For NP problems, finding the exact solutions is intractable. The approximate methods generate high quality solutions in a limited

amount of time, but they are not guaranteed the optimality of solutions (Russell & Norvig, 2010).

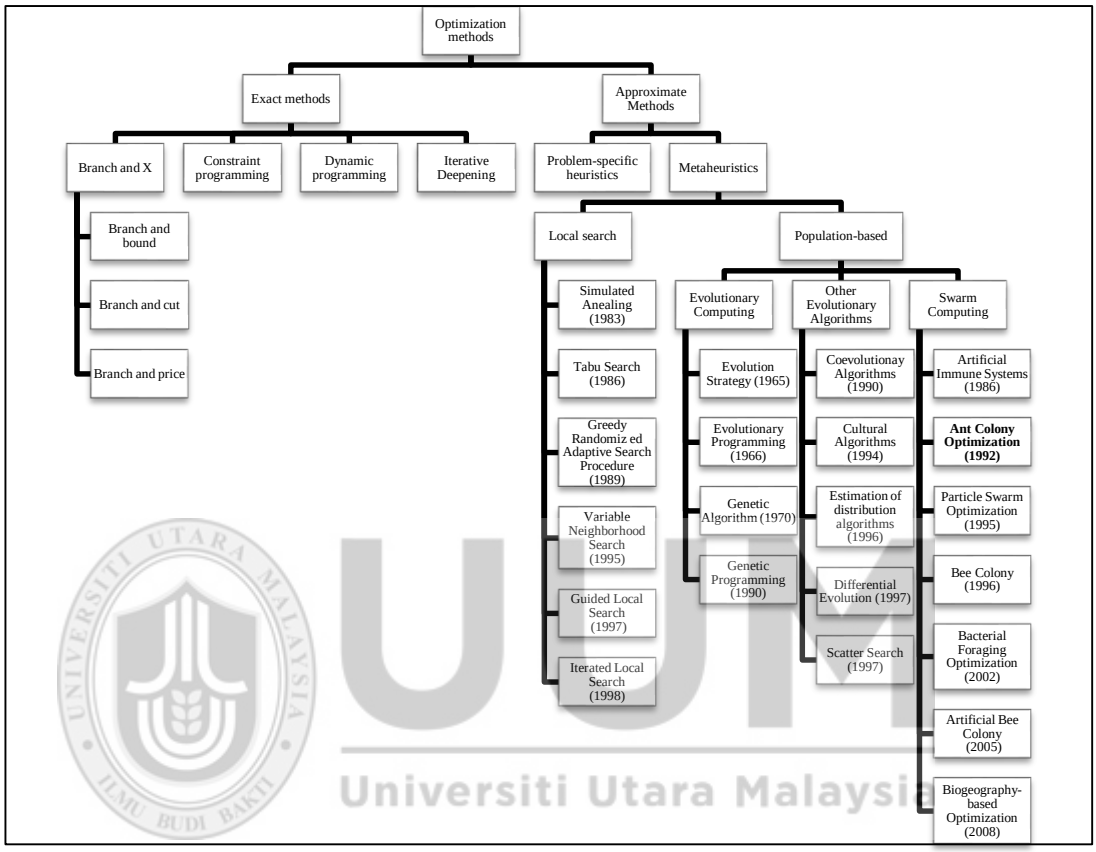


Figure 2.1. Classical Optimization Methods

Exact methods can be applied for CO problems of small size and simple structure. Following the size or the structure alone may not give much effect for applying the exact method. For example, some problems have a small size but their structure is very complex and vice versa. Another option is to implement this method in a large network of workstations (e.g. grid computing platform). In all cases, the exact methods must enumerate all the solutions of the search space and generate the optimal solution at the end.

In the class of exact methods, one can classify them according to their development community or their way of solving problems. Following the first division, dynamic programming and branch and X are developed in operations research community. While iterative deeping and constraint programming are developed in artificial intelligence community (Russell & Norvig, 2010). On the other side, following the second division, dynamic programming divides recursively the problem into subproblems based on the the principle that says "the subpolicy of an optimal policy is itself optimal". Another way of problem solving is by representing the problem as a search tree (e.g. branch and X; and iterative deeping methods). The root of the tree is the problem itself and the leaf nodes are its solutions. Finally, the optimization problem can be modeled as a set of variables connected by a set of constraints. This way of problem solving is called constraint programming. With large size or complex problem instances, the optimality of solutions, guaranteed by exact methods, will be sacrificed by finding (near-) optimal solutions.

In approximate methods, the general principle of applying this class of methods is to find good solutions for large size/complex problem instances in reasonable cost. They are classified according to their applicability into specific heuristic and metaheuristic methods. Unlike metaheuristics, the specific heuristic methods are designed to solve specific problems. Moreover, in practice, they are not useful for real life problems due to their way of guiding the search. They are more likely to fall in local optima, when the algorithm wastes the run time in unpromising regions of search space. In contrast, metaheuristics are general purpose methods. They are designed to escape from the local optima by using high level mechanisms.

Metaheuristics are algorithms designed to solve approximately a wide range of hard optimization problems without having to deeply adapt to each problem (i.e. general purpose algorithms). Indeed, the Greek prefix ‘‘Meta ’’, present in the name, is used to indicate that these algorithms are ‘‘higher level’’ heuristics, in contrast with problem-specific heuristics. They are classified, based on whether they manipulate a single solution or a collection of solutions at each stage, into local search metaheuristics and population-based metaheuristics (Boussaïd et al., 2013). In local search-based metaheuristics, a single solution is manipulated during the search, while in population-based ones, a whole population is involved.

These two metaheuristics' families have complementary characteristics. These are the local search methods which tend to intensify the search in local regions; they are exploitation-oriented. The population based methods allow diversifying the whole search space; they are exploration-oriented. Blum and Roli (2003) proposed a generic frame to understand how exploration versus exploitation is managed in metaheuristics (see Figure 2.2).

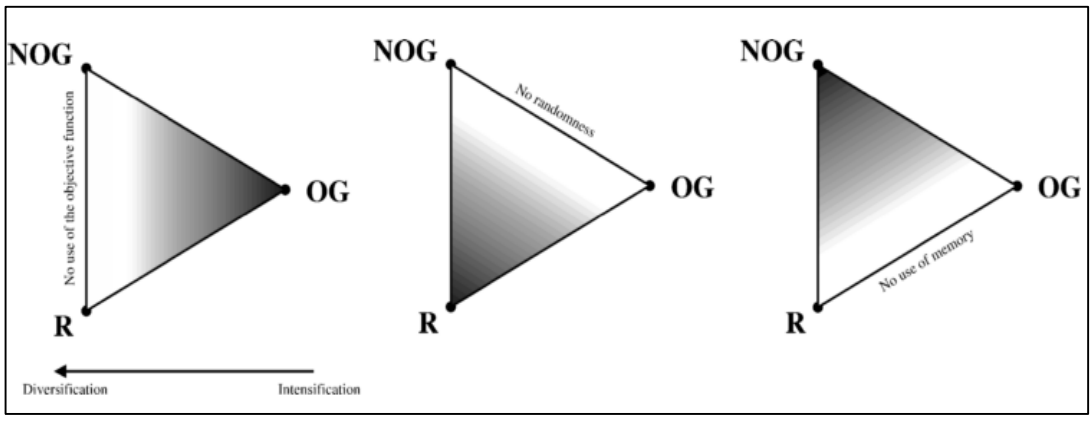


Figure 2.2. The Exploration and Exploitation Frame

Each component can be located somewhere on or in between the three corners of that frame, namely OG= objective function guided, NOG= nonobjective function guided, and R=randomness. For example, the basic exploration and exploitation components in ant colony optimization are pheromone update and probabilistic construction, while in tabu search, it is the neighbor choice (tabu lists) aspiration criterion. In the first example, the pheromone update is the exploitation component that is guided by an objective function. This component is influenced by an evaporation mechanism. The pheromone update component can be found on the line between NOG and OG. In the second example, when the tabu list (a NOG component) is long, the search will be exploration-oriented, i.e. close to the corner R.

The popularity of metaheuristics has been increased through their successful application to a large number of domains such as: engineering design, topology optimization and structural optimization in electronics, aerodynamics, fluid dynamics, telecommunications, automotive, and robotics; machine learning and data mining in bioinformatics and finance; system modeling, simulation and identification in chemistry, physics, and biology; control, signal, and image processing; planning in routing problems, robot planning, scheduling and production problems, logistics and transportation; and supply chain management (Talbi, 2009).

The growing complexity of real-world problems has motivated metaheuristic designers to search for efficient problem-solving methods. Divide and conquer techniques are one way to solve large and difficult problems. Division of large work

into smaller parts and combining the solution of small problems to obtain the solution of large ones have been a practice in computer research since long ago. Swarm also exhibits the behavior of division of work and cooperation to achieve difficult tasks. Swarm intelligence metaheuristics are outstanding examples which show that nature has been an unending source of inspiration (Manju & Kant, 2013). In particular, ants have inspired a number of methods and techniques among which the most studied and the most successful is the general purpose optimization technique known as ant colony optimization (Dorigo & Stützle, 2010).

2.3 Ant Colony Optimization

Ant colony optimization (ACO) takes inspiration from the foraging behavior of some ant species. These ants deposit pheromone on the ground in order to mark some favorable path that should be followed by other members of the colony. Ant colony optimization exploits a similar mechanism for solving optimization problems.

2.3.1 Biological Inspiration

A colony of artificial ants and its characteristics are inspired by the real ants' foraging behavior. Real ant foragers are traveling all the time to find food sources. Table 2.1 represents an emulation of the food foraging behavior and how it can be transformed to a CO problem (e.g. TSP).

Table 2.1

Artificial Ants versus Real Ants

Real Ant	**Artificial Ant**
1. Food sources	*Problem solutions (the routes in TSP).*

2. *Foraging* to find and exploit the nearest food sources to the nest.	*Searching*: exploring and exploiting the best solutions of the combinatorial problem.
3. Lives on the ground environment where the time not considered.	Associated with bi-dimensional grid termed construction graph.
4. *Pheromone* is a chemical substance that ants lay it on the ground during their foraging. Its density is directly proportional to the quality of food.	*Artificial Pheromone* is a numerical values assigned to the problem states during search. Its value inversely proportional to the quality of solution (the shortest distance).
5. *Stigmergy or multirenewal:* indirect communication among ants when one of them changes the environment (laying pheromone) and the others make use of this change later (following that pheromone).	*Multiple communications:* several artificial ants iteratively search based on the traveling salesman route. Each one behaves separately to construct its own route. At the end of the iteration, all ants have to finish constructing their own route.
6. *Mass Recruitment* (*or Pheromone trail*) is a chemical trail of pheromone.	*Artificial Pheromone Trail* is a vector of numerical values. Each trail represents a particular solution.
7. *Food Recruitment* is the process whereby the ants are influenced by each other through using pheromone.	*Trail Reinforcement* is the process whereby the artificial ants learn from each other.
8. *Autocatalytic behavior* (i.e. positive feedback) is a collective behavior where the more ants follow the trail, the more attractive that trail becomes.	Apply the following *stochastic rule*: the probability that an edge in a construction graph is included into the ant route is proportional to its pheromone value and heuristic value.
9. *Negative feedback* is the process of limiting the positive feedback. It may result from the limited number of ants, the food source exhaustion, and the evaporation of pheromone or a competition between paths to attract foragers.	Using *evaporation rule* to avoid *stagnation*. The evaporation should not be fast to prevent forgetting the previous experience and break down the cooperative behavior of artificial ants.
10. *No memory*	Has some memory called *tabu list*.
11. *No visibility*	*Visibility* is a static quantity derived from the distance between cities. It represents the heuristic desire to visit the next city.

2.3.2 Problem Representation

The ACO metaheuristic is based on a generic problem representation and the definition of the ants' behavior (Dorigo & Stützle, 2004). Given this formulation, the

ants in ACO build solutions to the CO problem by moving concurrently and asynchronously on a predefined construction graph. Considering the CO problem (i.e. TSP) as defined in Section 2.1, there are some aspects that need to be characterized: i) a finite set of components of the problem: C= $\{c_1, c_2, \ldots, c_n\}$, where n is the number of components of the TSP problem; ii) a sequence of the states of the problem over the elements of C, such that each sequence is S=$<c_i, c_j, \ldots, c_h, \ldots>$, and the set of all sequences is denoted by S; iii) a set of candidate solutions S^* is a subset of S; iv) a set of feasible solutions N is a subset of S; v) a non-empty set of optimal solutions; and vi) a cost g (s, t) is associated with each candidate solution.

TSP (S, f, Ω) has to be mapped to a complete connected graph called construction graph CG_{TSP} = (C, L), where C is the set of nodes of the graph and L is the connections of those nodes. The artificial ants will walk randomly on CG_{TSP} to build solutions of the TSP problem. The pheromone trail value τ and heuristic value can be associated with C or L.

2.3.3 The ACO Metaheuristic

ACO has been formalized into a metaheuristic for solving CO problems (Dorigo, Di-Caro, & Gambardella, 1999; Dorigo & Di-Caro, 1999). A series of generic guidelines allows a boost in the use of ACO methodology for problem solving (Monteiro, Fontes, and Fontes, 2012). Firstly, a finite set C of solution components needs to be derived. This set C is used to assemble solutions for the CO problem. Next, a set of pheromone values τ is defined. The set τ is called the pheromone model and is commonly recognized as a parameterized probabilistic model. The

pheromone model is probabilistically used to generate solutions based on the solution components. To achieve this, the model associates the solution components to the pheromone values $\tau_i \in \tau$ which forms the central components of the ACO metaheuristic. In general, the ACO approach attempts to solve an optimization problem by iterating the following two steps: i) construct candidate solutions by using the pheromone model. The pheromone model, as mentioned earlier, is a parameterized probability distribution over the solution space; and ii) modify the pheromone values by using candidate solutions in a way that it is deemed to bias future sampling towards high quality solutions (Blum, 2005a). The interaction between the two steps is presented in Figure 2.3 which illustrates a conceptual abstraction about how the ACO metaheuristic solves the CO problems.

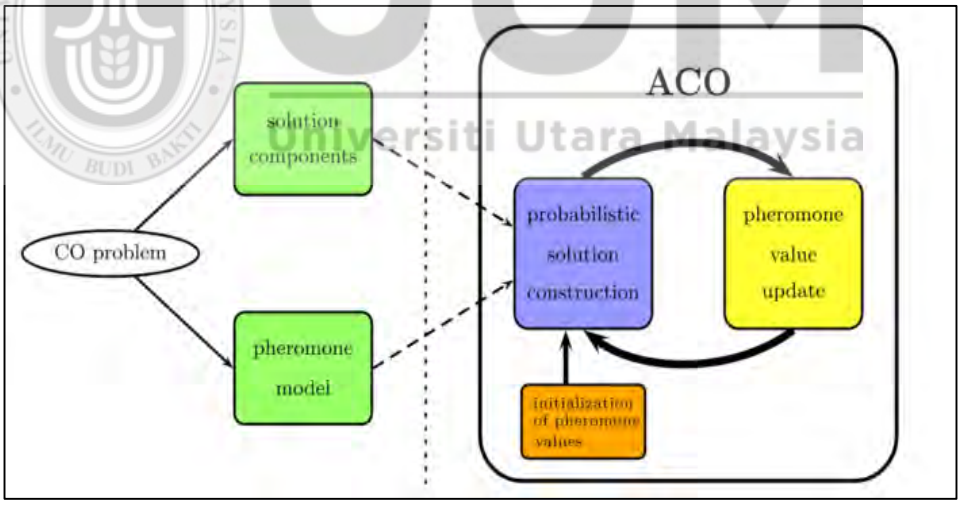


Figure 2.3. The Conceptual Framework of ACO

The ACO metaheuristic defines the way of the solution construction, the pheromone update, and possible *daemon actions*. These are used to implement specific problems or centralized actions that cannot be performed by a single ant (Cordon, Herrera, &

Stützle, 2002). An informal high-level description about ACO metaheuristic functionality is given in Figure 4.2.

```
1 Procedure ACO_Metaheuristic
2     parameter_initialization
3     while (termination_criterion_not_satisfied)
4         schedule_activities
5                 ants_generation_and_activity()
6                 pheromone_evaporation()
7                 daemon_actions() {optional}
8         end schedule_activities
9     end while
10 end Procedure
```

Figure 2.4. The ACO Metaheuristic Pseudocode

Parameter initialization: At the start of the algorithm, parameters are set and all pheromone variables are initialized to a value $\tau_{i=0}$, which is a parameter of the algorithm.

Ants' generation and activity: The ants build solutions to the CO problem by traversing nodes of the construction graph one after another until it finishes constructing complete solutions. Each ant, to move to the next node, applies a stochastic mechanism, which is biased by the pheromone and heuristic values. Optionally, an ant deposits/releases some pheromone at the visited nodes stepby step (online step-by-step pheromone update) or delays it until it constructs the current solution (online delayed pheromone update).

Pheromone evaporation: The aim of this activity is to decrease the pheromone values associated with all solutions. This mechanism is triggered by the environment

and it refers to the food exhausting in nature, while in the algorithm, it is used to allow the ants to explore new space regions.

Daemon actions: This part contains all the centralized procedures, which cannot be performed by a single ant such as global pheromone update. Usually, the daemon action replaces the online delayed pheromone update with the offline delayed pheromone update.

According to the way of updating pheromone, three algorithms have been developed: ant-density, ant-quantity and ant-cycle (Dorigo et al., 1991). The former two algorithms used the online update while the later used the offline update. Preliminary experiments run on a set of benchmark problems have shown that ant-cycle's performance was much better than that of the other two algorithms. This is the AS algorithm (Dorigo, Maniezzo, & Colorni, 1996; Dorigo, 1992). It was the basic model for subsequent successful ant algorithms. The other two algorithms (i.e. ant-density, ant-quantity) were abandoned.

2.3.4 The First Ant Algorithm: Ant System

This algorithm was the result of several experiments on the real ant foraging behavior. The experiments were conducted by Deneubourg et al. (1990). In AS, the main algorithmic components are outlined as follows:

Construct ant solutions. Solutions assemble as a sequence of solution components $C= \{c_{1,\ldots,}c_n\}$, which is derived from the problem under consideration. In the case of TSP, each edge of the TSP graph represents a solution component. Ant (k), moving

from its current city (i) to the next city (j), will construct one step in its own solution. This solution starts with an empty sequence s = ‹ › and will be extended in each construction step, by adding new feasible solution components from the set N (s^p) ⊆ C \ s. In each construction step, an ant chooses the next city stochastically through the following probabilistic decision (i.e. state transition) rule (Dorigo & Stützle (2010).

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}.\mu_{ij}^{\beta}}{\sum_{c_{il}\in N(S^P)} \tau_{il}^{\alpha}.\mu_{il}^{\beta}} \; if \; c_{il} \in N(S^P) \\ 0 \quad otherwise \end{cases}, \tag{2.1}$$

where τ_{ij} is the pheromone value adjusted by the parameter α and μ_{ij} is the heuristic value, which is given by: 1/distance (i, j). μ is adjusted by the parameter β. The specification of N (s^p) depends on the solution construction mechanism. In TSP, the solution construction mechanism restricts the set of traversable edges (i.e. N (s^p)) to the set of untraversed edges by ant (k).

Update pheromones. After the ants have built their tours and before the ants start to deposit pheromone, pheromone evaporation on all arcs is triggered. The main role of evaporation is to avoid too rapid convergence of algorithm (i.e. stagnation). It implements a useful form of forgetting the past history and focusing on new promising areas in the search space. Then, the ants deposit pheromone on pheromone trail variables associated to the visited arcs to make the visited arcs become more desirable for future ants. The updating phase is conducted through the following rules (Dorigo & Stützle, 2010).

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} , \tag{2.2}$$

where the $\Delta\tau_{ij}^{k}$ is determined by:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & if\ ant\ (k) used\ edge\ (i,j) in\ its\ tour, \\ 0 & otherwise, \end{cases} \quad (2.3)$$

where, Q is a constant and L_k is the length of the tour constructed by ant (k), while parameter $\rho \in [0, 1]$ is a pheromone trail decay coefficient (i.e. evaporation rate). Once the ants finish the pheromone updating, they will die (i.e. current iteration has been finished).

The amount of pheromone trail τ_{ij} (t) associated to the arc (i, j) is intended to represent the learned desirability of choosing city j when the ant is in city i. The pheromone trail learning changes during the problem solving to reflect the ants' experience with the problem search space. The pheromone amount deposited ($\Delta\tau_{ij}^{k}$) is inversely proportional to the quality of solutions (i.g. the shortest paths in TSP) the ants produced. This will direct the search toward good solutions.

The memory of each ant is represented by what is called the tabu list, which contains the already visited cities. The memory is used to define, for each ant (k), the set of cities that an ant located on city i (i = 1, 2, 3,…, n) still has to visit. By exploiting the memory, an ant k can build feasible solutions (in the TSP, this corresponds to visiting a city exactly once). Furthermore, the memory allows the ant to cover the same path and apply online delayed pheromone update.

The stochastic way in which the pheromone update prevents ants from ever reaching the optimum solution because it makes them reproduce the same solution, which

known as the *stagnation problem.* Even though the original AS algorithm achieved encouraging results for the TSP problem, it was later found to be inferior to state-of-the-art algorithms for the TSP as well as for other CO problems. Therefore, several variants of AS have been proposed in order to improve its performance. Figure 2.5 shows the chronological development of ACO metaheuristic framework over the years as drawn for this research.

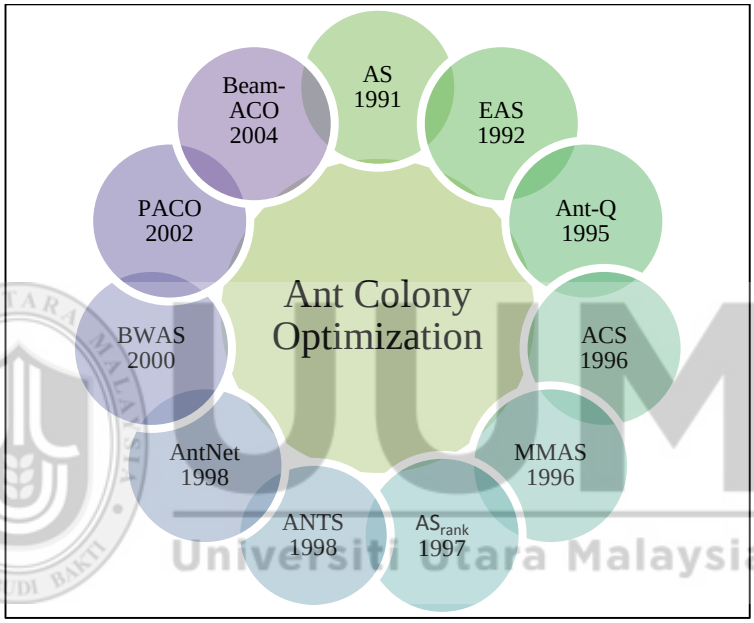


Figure 2.5. The Evolution of ACO Algorithmic Framework

The first improvement in ACO, called the elitist ant system (EAS), consists in depositing additional pheromone using the best-so-far tour. Other improvements were ant based Q-learning algorithm (Ant-Q), ant colony system (ACS), MMAS, and rank-based ant system (RAS). The Ant-Q was intended to create a link between ACO and reinforcement learning. The pheromone update rule of Ant-Q has been simplified to produce ACS algorithm. It is for this reason that Ant-Q was abandoned while ACS was restrained. In MMAS, the pheromone trail strength has been bounded to the interval [t_{min}, t_{max}]. The pheromone update in MMAS and ACS are

performed using the elitist strategy. AS_{rank} extends the elitism by using ranking strategy: it sorts the solutions according to their quality and the high ranked and elitist solution utilized in the pheromone update. The approximate nondeterministic tree search algorithm (ANTS) exploits the idea of lower bounds (LB) on the completion of a partial solution which is derived from branch-and-bound. The heuristic information, by means of lower bounds, will be computed to be used by each ant during the solution costruction. The objective function is more dynamic because it depends on the difference between LB and the ant's solution qualities. The heuristic information in the ACO algorithm for data network routing, denoted as AntNet, is more reactive than ANTS. It depends on the instantaneous state of the node's queues (i.e. the queue waiting time). The reinforcement learning of AntNet is based on the functions of the goodness of the ant's path (i.e. ant's trip time) and of the goodness of some relative measures which depends on the traffic conditions. In best-worst ant system (BWAS), while the best ant is utilized again to deposite pheromone (the elitisim), the worst ant also allows to subtract the pheromone. For dynamic optimization, population-based ACO (PACO) has been proposed. While the algorithm stems the max-min bounding from MMAS, it proposes a new pheromone deposite/subtract mechanism called FIFO-Queue. In beam search-based ACO (beam-ACO), the beam saerch stems the lower bound concept as in ANTS. Beam-ACO uses parallel exploration by taking *n* nodes of search tree and expanding them in *m* direction based on the LBs; this results *n*m* partial solutions.

According to Dorigo and Stützle (2010), the substantial difference among ACO variants is in the way of guiding the search. In other words, it is due to the way of

managing the exploration of the search space and the exploitation of the best solutions found.

2.4 The Max-Min Ant System

The MMAS algorithm is achieved by the strongest improvement over ACO variants for the TSP and it is among the best available algorithms for the QAP (Stützle & Hoos, 2000). The main modifications of MMAS are the continued use of elitist strategy; limiting the stagnation of the search by τ_{max} and τ_{min} bounds ; using τ_{max} to initialize the trails, and finally smoothing (or restarting) the trails to stop the stagnation and to increase diversification feature (Stützle & Hoos, 1998). The modifications are discussed in more detail as follows.

2.4.1 Pheromone Trail Update

The MMAS uses only one ant to update the pheromone. This ant is called the best ant. For this choice, there are two possibilities: use the iteration-best ant or the global-best ant. The first technique is the best for long-term runs where the diversification aspect will be high. On the contrary, the second one is useful in short-term runs with the risk of entering a stagnation situation. Allowing more ants to update the trails is the major following in ACO (e.g. ranking and generalized elitism strategies) (Stützle & Hoos, 1998). The modified updating rule relies on several feedback measures. At the start of the algorithm, the iteration-based update rule is used more often, while during the run of the algorithm, the frequency with which the global-based update is used increases. Following this technique will not guarantee

that the same solution components will not be reproduced by Equation (2.1) (Dorigo & Blum, 2005). In such situation, if it occurs for all nodes, the search stagnates.

2.4.2 Pheromone Trail Limits

The MMAS algorithm involves the pheromone trail limits to avoid the worst case of this phenomenon; it introduces pheromone trail limits to influence the selection probability of the respective solution component. The selection probability is determined by pheromone trails and heuristic information. Limiting the extreme values of pheromone trail can be denoted by $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. The parameter τ_{min} has the main influence in avoiding the stagnation. The parameter τ_{max} is still useful in the initialization phase. Initializing the pheromone trails with maximum values allows high exploration (Dorigo & Stützle, 2004). According to Dorigo, Birattari, and Stützle (2006); and Stützle and Hoos (2000), both τ_{min} and τ_{max} values are typically obtained empirically and tuned on the specific problem at hand. Nonetheless, some guidelines have been provided for defining τ_{min} and τ_{max} on the basis of analytical considerations. In particular, using the lower trail bounds improves the performance of MMAS compared with the quality of solutions of MMAS without using these limits.

2.4.3 Pheromone Trail Restart

The MMAS algorithm uses the restart, or so called re-initialization, whereby the pheromone trail by MMAS sets the value of pheromone deposited on all arcs to the maximum possible trail strength, i.e. to the quantity τ_{max}. This type of setting will increase the initial exploration of the algorithm. To illustrate the usefulness of this

setting, the following situation will be considered. In the first iteration, the pheromone trails will be decreased according to the evaporation factor as follows: τ_{ij} (t+1) = ρ. τ_{ij} (t). Hence, the relative difference among pheromone trails will be as follows: ρ, ρ^2, etc. On the contrary, if the lower pheromone trail is utilized for initialization, the difference among pheromone trails will increase more strongly. Thus, the selection probabilities according to Equation (2.1) will evolve more slowly when initializing the trails to τ_{max} and, hence, the exploration of solutions is favored.

2.4.4 Pheromone Trail Smoothing

The pheromone trail smoothing (PTS) mechanism is designed in MMAS to increase the robustness of using these limits, especially the lower trail limits. This mechanism is known by smoothing the trails and is used to counteract the stagnation of search for long term running. This mechanism can be interpreted as an urgent update in order to produce new tours by influencing the probabilistic distribution for the tour construction stage in the next iteration. When MMAS has converged or is very close to convergence (as indicated by the average branching factor), this mechanism increases the pheromone trails proportionally to their difference to the maximum pheromone trail limit. The proposed mechanism has the advantage that for δ < 1, the information gathered during the run of the algorithm (which is reflected in the pheromone trails) is not completely lost, but merely weakened. For δ = 1, this mechanism corresponds to a re-initialization of the pheromone trails, while for δ = 0, PTS is switched off (Stützle & Hoos, 1998, 2000).

2.4.5 Pheromone Trail Learning

After each iteration, the pheromone trail level on all arcs will be decreased. On the other side, the good arcs crossed by the best ants will be maintained to keep the high level of trail strength according to the reinforcement rule. Distinction between the good arcs and bad arcs during the run is interpreted in MMAS as a learning process (Favaretto, Moretti, & Pellegrini, 2009; Pellegrini, Favaretto, & Moretti, 2006; Stützle & Hoos, 1998). The evaporation rate ρ is played as the main rule in this process. The speed of learning depends on the value of ρ. There are two possibilities to assign ρ's value: high and low values. With the high values, the speed of learning will be slow, so the algorithm needs to spend more iterations (i.e. more time to learn which arc is good) to do that. While with low values of ρ, the learning speed will increase, so that, with long and short-term runs, the low values of ρ are more preferable.

2.4.6 Hybridizing with Local Search

Although MMAS can be applied without coupling with local search procedures, very often its solutions' quality is greatly improved if it is extended to include it. The first step in applying local search is the definition of a neighborhood structure over the set of candidate solutions. One common way of defining neighborhoods is via k-exchange moves that exchange a set of *k* components of a solution with a different set of *k* components. This kind of local search called k-opt neighborhood. Three k-opt heuristics have been implemented to improve the quality of solutions, they are 2-opt, 3-opt and 2.5-opt (Johnson & McGeoch, 2007). The 2-5-opt is a restricted

version of 3-opt, where the segment of solutions that contains only one component is considered to check whether the exchanges result in an improved solution or not.

From the exploration and exploitation point of view, the pheromone trail management plays the main role in changing the probabilistc distribution of search space (Dorigo & Stützle, 2010). If the pheromone concentration is high, then the probabilistic construction will tend to be aggressive and the exploration amount will be low. Both probabilistic construction and pheromone update components in ACO are guided by memory-based strategies.

2.5 Memory-based Strategies for Exploration and Exploitation

The rationale work for managing E&E in ACO is about how a memory model called pheromone is managed. The E&E remains the base for problem solving by search algorithms. According to Beer et al. (2012), "exploration refers to how widely an algorithm surveys the search space". Beer et al. also defined the exploitation as "the speed at which the algorithm converges to a local minimum and is related to pheromone." The study also showed that if exploration takes precedence, the algorithm will explore unproductive areas of the search space before reaching a solution; if exploitation is too strong, the algorithm may converge prematurely and produce a poor result (Beer et al., 2012). Dorigo (1992), in his Ph.D thesis, discussed different ways to achieve such balance. One of them is by sampling a good solution with the probability of one, while the bad solution with zero. This way of representation reflects the main drawback, which is the early stagnation. Alternatively, a simple way to do that is by exploiting the quality of solution of each

ant as a function of updating pheromone (Dorigo, 1992). Several memory-based E&E forms exist in the literature which can be classified as follows.

2.5.1 Quality-Dependent Strategy

This strategy first appeared in the AS algorithm as a solution to prevent stagnation (Dorigo et al., 1996). In this strategy, some measures utilize the quality of solution generated to calculate the amount of pheromone that should be deposited by ants. The main drawback of this strategy is its poor performance. As a result, the need to seek alternative ways to enhance the performance of ant algorithms arises. Talbi, Roux, Fonlupt, and Robillard (2001) extends this strategy in a similar way by allowing all ants to deposit pheromone, where the difference between the current solutions F(S) and the worst ones F(S-) and its proportionality to the best solution F(S*) represent the amount of pheromone to be used in the reinforcement process. It is also similar to the best-worst strategy in BWAS (Cordon, Herrera, & Moreno, 2000) in exploiting the best and worst ants. The pheromone quantity laying on the solution components in this strategy may formulate differently. Following the Hyper-Cube Framework proposed by Blum, Roli, and Dorigo (2001), Blum and Blesa (2005) proposed an ACO algorithm to solve edge-weighted k-cardinality tree problems, where the pheromone reinforcement uses the convergence factor to calculate the quantity of deposited pheromone instead of using the objective function. Shyong, Pengyeng, and Bertrand (2004), in their ACO method to solve the minimum weight vertex cover problem (MWVC), proposed another formulation to derive the objective function, which is by using total weight of the nodes in the solution to calculate the quantity of the deposited pheromone in each iteration.

Solimanpur, Vrat, and Shankar (2005) used a scaling factor λ to adjust the amount of pheromone laid on the componenets of solutions F(Sk) that are closest to the global best solution F(S*). This method clearly avoids full convergence and encourages search along the vicinities of the global best solution in the hope that a better one can be found nearby. Bin, Zhongzhen, and Baozhen (2009) and Zhongzhen, Bin, and Chuntian (2007) have proposed a new strategy to update the increased pheromone, called ant-weight strategy, by which all ants are allowed to update their paths locally and globally. A solution is only entirely defined when all routes constructed are assembled. There are also two kinds of pheromone increments: local and global. The local pheromone increment uses the contribution of each arc to the prespective tour, which increases when a specific indicator decreases. The global one uses the total length of solution and the number of tours. Since the solution quality is the only trigger to decide how much the solution should be awarded, its effectiveness in dynamic environment needs to be improved. For example, in dynamic TSP some of the components of current solution may vary before the solution is evaluated. This mislead the way in calculating its award on the assumption that its quality will be improved.

2.5.2 Quality-Independent Strategy

This strategy is applied in geographically distributed problems, like network problems, where the pheromone deposited by ants is equal to constant as in the simple-ant colony optimization algorithm (S-ACO). As in the case of real ants, autocatalysis and differential path length (DPL) are at work to favor the emergence of short paths (Dorigo & Di-Caro, 1999). Di-Caro and Dorigo (1998), in AntNet

algorithm, depended on this autocatalysis property. In network communication, where AntNet is applied, the environment changed so the pheromone amount deposited by the ant is switched off. Shyong et al. (2004), in their ACO method to solve the minimum weight vertex cover (MWVC) problem, stated that the solution does not necessarily constitute a path or a tree on the underlying graph; the objective function is a constant value (τ_0). This fomula is equivalent to the local pheromone edupdate in ACS algorithm (Dorigo & Gambardella, 1997). In short, evaluate the solutions independently of their quality of solutions is more analogous to the behaviour found in real ant colonies. However, it is not a recurring theme in ACO research especially in static environments where the quality-based evaluation is the dominant theme as it can accelerate the convergence of ants toward optimal or near optimal solutions.

2.5.3 Elitist Strategy

A first improvement over AS was obtained by the elitist ant system (EAS) (Dorigo, 1992). The basic idea of this strategy is to provide an additional reinforcement to the best-so-far solutions. This strategy is hereby to increase the exploitation by introducing a strong bias towards the solution components of the best-so-far solutions. Another elitist variation has been studied in MMAS and ACS, where either of the iteration-best solution or the best-so-far are used for elitist reinforcement. Maniezzo (1999), in ANTS algorithm, proposed a new formula to calculate the pheromone amount to be used in the reinforcement process and to determine the elitist solutions (L_{avg}). L_{avg} is the moving average of the last l solutions, that is, it is the average length of the l most recent tours generated by the

algorithm. If an ant's solution is worse than the current moving average, the pheromone trail of the arcs used by the ant is decreased; if the ant's solution is better, the pheromone trail is increased.

Blum (2002) proposed a list (L_{elite}) to record the elite solutions found during the search (i.e. iteration-best solutions, best-so-far soluions and restarting best solutions) to be used in reinforcement. Following this work, Blum and Blesa (2005) proposed an ACO algorithm to solve edge-weighted k-cardinality tree problems, where the pheromone reinforcement uses three kinds of weighted solutions: best iteration solution S_k^{ib}, best global solution S_k^{gb} and best restart solution S_k^{rb}. In a different way, Rappos and Hadjiconstantinou (2004) developed a new approach considering the nature of the designing flow networks problem. The objective function for each edge consists of a fixed component and a variable component. Consequently, the authors decided to use two kinds of pheromone trails to be deposited: fixed trail Te (ij) and variable trail Tf (ij). There are also two kinds of ants: reliable ant and flow ant. Only flow ant is allowed only to reinforce both trails. Whereas the second ant is used to add a reliability arc to that solution produced by the flow ant.

Ku-Mahamud and Alobaedy (2013) used two kinds of trail reinforcement during the run of the algorithm: pheromone reinforcement and heuristic reinforcement, where the second one is triggered to reflect the first one. The concept is that when the best-so-far ant globally updates their trail, the environment will be changed, and thus a new heuristic value will be obtained. This enhancement affects the convergence behavior of ACS in its application to two NP-hard problems: TSP and Job

Scheduling in Grid computing and produced good results. Alaya, Solnon, and Ghedira (2004) calculated the objective function such that the pheromone amount deposited on the solution components is inversely propositional to the difference between iteration-best solution and the best-so-far solution. Therefore, the closer the solution is to the global best solution, the higher the quantity of pheromone is deposited.

The concept of *elitism* is inspired from the genetic algorithm (GA), where the best solution is found in the current iteration corresponding to the fittest individuals of the current generation (Goldberg & Holland, 1988). The main drawback of this strategy is that the local search aspects represented by exploitation will be more important than the global search aspects represented by exploration, and that contrasts with the idea of inventing metaheuristics.

2.5.4 Rank-Based Strategy

Another improvement over the elitist strategy is the rank-based strategy, which was first proposed in RAS algorithm (Bullnheimer, Hartl, & Straub, 1997). The idea of ranking is inspired also from the genetic algorithm field: first, the population is sorted according to fitness, and then the probability of being selected depends on the rank of an individual. The ranking strategy is used to counteract the shortcoming of the elitist strategy. The contribution of AS_{rank} is obtained by sorting tours constructed by m ants by their lengths and to be weighted then according to their rank in that sorted list. The pheromone amount will be deposited according to the rank of the ant, while only the m^* best ants are considered for reinforcement. With ranking

strategies, the balance between exploration and exploitation can be achieved. Comparing with elitist versions of ant systems, rank-based version is slightly better results. However, it likely to stuck in local optima which demonstrates the stagnation behaviour especially with large scales environments.

2.5.5 Trail Learning Strategy

The trail learning strategy allows stronger exploitation to be achieved by applying the new aggressive decision rule of ACS algorithm (Dorigo & Gambardella, 1997) and by applying Q-learning of Ant-Q (Gambardella & Dorigo, 1995). An important contribution in ACS is in the decision rule used by the ants during the construction process; so-called *pseudo random proportional* rule. In which, the probability for an ant to move from city i to city j depends on a random parameter q, which is uniformly distributed over [0, 1], and a parameter q_0. Where, if $q \leq q_0$, then j = argmax $_{cij \in N\ (sp)}$ $\{\pi_{ij}{}^{\alpha}. \eta_{ij}{}^{\beta}\}$ (i.e. a biased exploitation: intensifies the knowledge available about the problem $\pi_{ij}{}^{\alpha}$ and $\eta_{ij}{}^{\beta}$), otherwise (if $q > q_0$), the decision rule which operates the transaction rule in AS is used (i.e. a biased diversification). Tuning q_0 allows to modulate the degree of diversification and to choose whether to concentrate the activity of the system on the best solutions or to explore the search space.

Blum and Blesa (2005) introduced some changes to the probability distribution rule defined for ACS (Dorigo & Gambardella, 1997) in order to solve k-minimum spanning tree problems. An ant starts its solution by randomly choosing the first arc to enter the solution tree. Then, at each step of the construction, the next solution

component is chosen deterministically if $q \leq 0.9$, and probabilistically if $q > 0.9$. This rule assigns equal weight to the pheromone and the heuristic values by eliminating parameters α and β from the exponents of the pheromone and heuristic values respectively. Given that the probabilistic rule is only triggered whenever a random number q > 0.9, the search for solutions in 90% of the cases usually concentrated on relatively good areas.

In Altiparmak and Karaoglan (2007), if the global best solution has not changed after 50 iterations, then 10% of the solutions reconstruct randomly to increase the diversification. Afshar (2005) proposed a new probability distribution rule for ACO algorithms. The strategy is defined to prevent a domination of the pheromone trails in the ants' decision, by incorporating an additive form instead of the usual multiplicative form. This way, the author expects both pheromone and heuristic information to have an active role in the decision. This new probability distribution rule comes with a modification of the heuristic value, which is a simple normalizing procedure in which every heuristic value will be between zero and one regardless of the problem size. The major drawback of this stratgey is that its aggressive exploitation entail a loss in effeciancy of exploration even when the exploration/exploitationan is configured fairly, e.g. assigning 50% to parameter q_0 in ACS.

2.5.6 Online-Offline Update Strategy

To balance the strong exploitation of pseudo random proportional rule, ACS introduced the local pheromone update (i.e. online update). It is performed by all the

ants after each construction step. Each ant applies it only to the last edge traversed. The main goal of the local update is to diversify the search performed by subsequent ants during iteration: by decreasing the pheromone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and, hence, to produce different solutions. This makes it less likely that several ants produce identical solutions during one iteration.

The offline pheromone update is applied at the end of each iteration by only one ant, which can be either the iteration-best or the best-so-far. However, the update formula is slightly different. The parallel local update and stochastic global update rules increase the cooperation rule among agents and maintain the balance between local search and the global one (i.e. exploitation and exploration respectively). The global search is maintained by local rule: the pheromone amount concentrates on the arc (i, j) because the aggressive construction rule (i.e. pseudo random proportional rule) will be eaten by the respective ant using the local update rule. On the other hand, the local search is maintained by the global search: the pheromone amount is reinforced on the arc (i, j) that only belongs to the global best solution to encourage other foraging ants to search near the good solutions (Dorigo & Gambardella, 1997; Dorigo, 2007)

The online-offline update is recurred strategy in most ACO literature, not only ACS. Shyong et al. (2004) used the local update in each construction step. Similarly, Eswaramurthy and Tamilarasi (2009) have also used the global and local updating rule, but considered arcs instead of nodes. They applied their approach on Job Shop

Scheduling Problem. This strategy can cause oscillations during the costruction of solution leading to a large fluctuations in its performance.

2.5.7 Best-Worst Strategy

This strategy is epitomized in BWAS (Cordon et al., 2000) as another improvement of that one in AS. It incorporates evolutionary computing concepts where it uses the same probability distribution rule and evaporation mechanism. BWAS is characterized in its *best-worst pheromone trail updating rule*, where the arcs belonging to the best solution is reinforced and the ones that belonged to the worst solution and not present before in the best solution is penalized. BWAS is further characterized by using *pheromone trail mutation,* where the diversity in the search process is introduced. Each pheromone trail associated with each arc is mutated with the probability P_m by adding or subtracting the mutation rate $\tau_{threshold}$ in each iteration. This rate is less strong in the early stages of the algorithm and stronger in the latter ones where the stagnation is stronger. Maniezzo (1999), in ANTS algorithm, used the best-worst concept as well. In his method, the pheromone trails that belong to a particular solution is increased or decreased based on the degree of best or worst of that solution compared with the average solution quality of the last *k* solutions. Guntsch and Middendorf (2002), in FIFO-queue ACO algorithm, utilized implicitly the best-worst idea to propose a fast pheromone update rule. After each iteration, the best-or-worst solutions need to be added or removed from the population solution, then only the pheromon trails that belong to them will increase or decrease. However, this model still suffers from the slow convergence and low

searching efficiency (Dorigo & Stützle, 2004). The good solutions may not be produced by the reinforcement learning of ants.

2.5.8 Bounding Strategy

In this strategy, bounding the small values of the pheromones in the components of the solution by a minimum bound (τ_{min}) and the extremely large values by a maximum bound (τ_{max}), was investigated. The minimum bound is used to avoid prohibiting the choice of those arcs with small values of pheromone, while the maximum bound is used to avoid choosing the extremely big values for pheromone which will lead to the construction of the same solution, over and over again.

The first application to this strategy was proposed by Stützle and Hoos (2000) in MMAS. The author defines the pheromone maximum bound based on the evaporation rate ρ and the cost of the best-so-far solution while the minimum bound is calculated based on maximum bound τ_{max} and the number of arcs (i.e. solution components) and the probability of finding a best solution. Therefore, whenever a new global best solution is found, τ_{min} must also be updated. The bounding strategy can be denoted by $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. Experiments have shown that the τ_{min} has the main force to avoid stagnation, while the τ_{max} is still useful to derive τ_{min} and is used in pheromone initialization. Both parameters τ_{min} and τ_{max} are adaptively changed during the run of the algorithm. Once a new solution is found by the ants, their values will be updated. The important thing is how to manage their values update during the run. The author found a strong relation among the two parameters and their influence on the E&E balance (Stützle & Hoos, 1999). A better balance can be

provided by controlling the tightness of the trail limits by introducing the parameter min-factor: $\tau_{min} = \tau_{max}$/ min-factor. Venables and Moscardini (2006) and Altiparmak and Karaoglan (2007) both used the same formula as in MMAS to find τ_{max}, but changed the way to find τ_{min}. It is easy to see that τ_{max} and τ_{min} are adaptively changed when a new best solution is found.

Blum and Blesa (2005) defined another limitation schema, where $[\tau_{min}, \tau_{max}] = [0.001, 0.999]$, and re-initializing schema, when the convergence factor becomes closer to 1. Bin et al. (2009) reformulated the two bounds according to the distance between the depot and the other customers which are denoted by d_{0i} in vehicle routing problem (VRP), where bounds have to initialize when the algorithm starts. It is worth to mention that this strategy recurred in several ACO approaches and not only in the standard ACO. For example, in PACO approach, the pheromone values are never zero because of the minimum limitation used (Guntsch & Middendorf, 2002). The shortcoming of this strategy is that the scope of its impact appears only in the final phase of search making it ineffective in short-run application such as querying.

2.5.9 Restarting/Smoothing Strategy

This adaptive strategy entail increase the exploration of search by redistributing the pheromone amount among the incident arcs. It relies on changing the pheromone distribution by resetting pheromone values when some triggers are reactivated. It is applied successfully to guide the search of ant colony based on tabu search algorithm (ANTabu) (Talbi et al., 2001), and MMAS algorithm. Blum (2002) introduced a

new re-initializing scheme for MMAS in his Hyper-Cube Framework (HCF) (Blum & Dorigo, 2004). Venables and Moscardini (2006) proposed a new restarting mechanism. If the percentage of τ_{min} arcs exceeds 50%, then all arcs in the pheromone matrix will reset to τ_{max} and bounds have to be updated according to the global best solution. Bui and Zrncic (2006) developed an approach for helping ants to recognize the bad arcs and the good arcs. They proposed a new way to calculate the initial pheromone τ_0, τ_{min} and τ_{max} and then used the value of (τ_{max} - τ_0) to reset the arcs that exceed τ_{max}, and the value of (τ_0 + τ_{min}) to reset arcs values that go under τ_{min}.

The pheromone amount can be redistributed by smoothing strategy: increasing the pheromone trails proportionally to their difference to the maximum pheromone trail limit (Stützle & Hoos, 1998, 2000). Experimental results showed that applying this strategy can lead to high quality solutions, especially in the long run of MMAS. However, its performance is highly dependent on parameterization. When its intermediator (δ) < 1, the information gathered during the run of the algorithm (which is reflected in the pheromone trails), is not completely lost, but merely weakened. For $\delta = 1$, this mechanism corresponds to restarting the pheromone trails, while for $\delta = 0$, the pheromone trail smoothing strategy is switched off.

2.5.10 Colony-Level Interaction Strategy

This strategy is used explicitly for parallel implementations and multiobjective optimization (Middendorf, Reischle, & Schmeck, 2000). However, its implicit goal is to achieve a fine-tuned balance between intensification and diversification by multiple colonies and interaction between them in some way (Blum, 2002).

Kawamura, Yamamoto, and Suzuki (2000) developed the first multi ant colony optimization algorithm, where several colonies work in multi-E&E levels and share their experience by exchanging information. Aljanaby, Ku-Mahamud, and Norwawi (2010) proposed interacted multi ant colony optimization algorithm (IMACO) based on ACS. This work exploited the ACS built-in E&E strategies from one side and benefited from MACO's (multi ant colony optimization algorithm) advantages over ACO. Rahmani, Dadbakhsh, and Gheisari (2012) enhanced this strategy by adding the repulsion mechanism to reduce the likelihood that all colonies' exchanging information uses only the optimal solution. The dependency on the number of ants which is the principle shortcoming of this way entails two major issues. First, the computational efforts will be increased to produce one population. Second, this approach will not perform efficiently when applied in stochastic local search algorithms considering that local search is fundamental for finding high quality solutions.

2.5.11 Population-Based Strategy

This strategy is designed for dynamic optimization problems by using the population to facilitate a faster pheromone update process than the standard way. Utilizing the population concept entailed keeping track of the good solutions (i.e. exploitation) and omitting the visited solutions by removing them from the pheromone matrix (i.e. exploration). Guntsch (2004) is the first to propose a population-based ACO approach (PACO) in his Ph.D thesis. Guntsch and Middendorf (2002) developed the first PACO algorithm which is known as FIFO-queue ACO algorithm. This strategy keeps the stronger options of the previous strategies, such as the trail learning, the

bounding and the best-worst, in the standard ACO. Despite that the real benefit of population-based strategy was postulated to dynamic optimization it opens the way for applying a new diversity preservation technique called *niching.* It is used in the PACO algorithm, particularly to solve multiobjective problems in a multiple area search space. Niching generally aims to achieve diversity of search focus (Angus, 2008). However, two disadvantages can be recognized in this strategy. Its pheromone update will follow the quality-independent approach and its design is not well suited for static optimization.

2.5.12 Hybridizing Strategy

This strategy places ACO in relations with other approximation methods to produce algorithms with new hybrid E&E mechanisms. For example, the hybridization of MMAS algorithm with local search (Stützle, 1999) and that of ACO with beam search (Blum, 2005b). The vast literature on ACO verifies obtaining high quality solutions when local search algorithms are coupled with ACO. For example, the implementation of k-opt heuristics with MMAS. The experiments (Stützle, 1999) showed that the quality improvement returned by 3-opt is better than that of 2-opt and 2.5-opt. However, the extra computational time required for 3-opt was not worth the small quality improvement yielded unless speedup techniques are included. The speedup techniques that are involved in the early implementation of ACO are avoiding the redundancy in search space, restricted *neighbor list* and *don't-look bits*. The recent implementations of ACO concern on how better integration between local search procedures and construction solutions is achieved (Gambardella, Montemanni, & Weyland, 2012).

Blum, Puchinger, Raidl, and Roli (2011) justified this hybridizing technique by saying, "Research in metaheuristics for CO problems has lately experienced a noteworthy shift towards the hybridization of metaheuristics with other techniques for optimization. At the same time, the focus of research has changed from being rather algorithm-oriented to being more problem-oriented. Nowadays, the focus is on solving the problem at hand in the best way possible, rather than promoting a certain metaheuristic." However, they cautioned the use of this type of strategy without clear guidelines by mentioning, "The research community should make an effort to move towards a sound scientific methodology consisting of theoretical models for describing properties of hybrid metaheuristics and using an experimental methodology as done in natural sciences." The exploration amount, therefore, is automated in a history-sensitivity way, i.e. *automated exploration*: the exploration is changed only when there is evidence that more or less exploration is needed. The amount of exploration is increased or decreased based on a feedback scheme.

2.6 Exploration Measures in ACO

Several existing techniques can be used to determine the amount of exploration and exploitation needed in an ACO algorithm. These techniques can also be applied to aid the balance between exploration and exploitation. Battiti et al. (2008) stated that, "An automated heuristic balance for the "exploration versus exploitation" dilemma can be obtained through feedback mechanisms, for example, by starting with intensification, and by progressively increasing the amount of diversification only when there is evidence that diversification is needed."

2.6.1 Distance of Solutions

This is the simplest indication that can utilize the calculation of the number of arcs contained in one solution but not in the other. This indicator focuses on the duplicated arcs between two solutions, and if the degree of similarity is high, then the exploration amount is low (Dorigo & Stützle, 2004). A disadvantage of this measure is that it is computationally expensive. There are $O(n^2)$ possible pairs to be compared and each single comparison has a complexity of $O(n)$.

2.6.2 Average Lambda-Branching Factor

This measure, introduced in Gambardella and Dorigo (1995), depends directly on the pheromone trail values, and it makes it more suitable for tracking the ant behavior while the computation is going on. This technique measures the diversity of the pheromone trail values more directly and it does not change much from iteration to iteration (Stützle, 1999). The branching factor is the first proposed indicator in ACO algorithms, and can be defined as follows. Let ph_max (i, j) and ph_min (i, j) be the maximum and minimum pheromone amount respectively of all the arcs that exit from node *i*, and let *d* be the difference between two amounts, i.e., ph_max (i, j) and ph_min (i, j). The branching factor of node *I* is the number of arcs that is greater than λ*d + ph_min (i, j), where $0 \leq \lambda \leq 1$. The average of lambda branching factor of all nodes gives an indication of the amount of exploration conducted by each ant. The disadvantage of this factor is its dependency on the value of parameter λ.

2.6.3 Entropy

In information theory, uses to measure the confusion degree based on the probability of random events (Wang, 2013). Entropy is recurring theme in parameter tuning of metaheuristics (Eiben & Smit, 2011). This diversity indicator firstly introduced by Pellegrini (2006) to analyze the behaviour of some ant algorithms against the change in parameters' values. Colas and Monmarch (2008) have applied this concept during the solution construction in order to adapt the search of ACO as follows. At each node, the ant will calculate the selection probabilities of all other nodes, i.e. entropy, according to the following rule. $\varepsilon_i = -\sum_{j=1}^{l} p_{ij} \log p_{ij}$ where p_{ij} is the probability of choosing arc (i, j) when being in node *i*, and *l*, $1 \leq l \leq$ n-1, is the number of possible choices. Therefore, finding the average of all entropies will be in the same way. Nevertheless, recruiting this formula in constructing solution is not efficient because it will complicate the calculations inside the colony.

2.6.4 Convergence Factor

The convergence factor was introduced by Blum (2002) to be used in hyper–cube framework. It is used for tracking what is called as the *extent of keep stuck*, which is the converging phase of all ants. Dorigo and Stützle (2004) mentioned it as a good way to calculate the amount of exploration using the rule: $\sum_{\tau_{ij} \epsilon T} min\{\tau_{max} - \tau_{ij}, \tau_{ij} - \tau_{min}\} / n^2$, where *T* is the pheromone matrix, and τ_{ij} is the pheromone trails. Seo (2009) recruited this measure in calculating how close the pheromone values on the updated arcs from the maximum and minimum threshold of pheormone. The major drawback of this measure is in its applicable for only max-min ant colony framework, which can be considered as a loss of generality.

2.6.5 Acceptance Criteria

Acceptance criterion is the condition under which the new local optimum can be considered satisfy to be replaced with the current one (Boussaïd et al., 2013). This tool has been used with the restart strategy to increase the exploration amount in MMAS and BWAS algorithms. It has been firstly introduced in ACO by Stützle (1999) in his thesis: if the i_{last} be the iteration counter i in which the best iteration solution has been found, and then restarting can be modeled by the acceptance criteria $Restart(s'', s''', history)$ where it equals to $s''\ if\ f(s'') < f(s)$; $s'''\ if\ f(s''\ ) \geq f(s)\ and\ i - i_{last} > i_r$ or S otherwise. Based on the formula above, the value of s''' is generated randomly by a new initial solution which corresponds to a restart of the algorithm. This formula is typically used in MMAS and BWAS. The amount of exploration will be increased without guarantee that the same regions are not visited again. Another weakness is the difficulty of decision-making during the run, i.e. whether to use extreme acceptance criteria (preferring exploitation) or to accept any criteria (preferring exploration).

2.6.6 Exploration Measure/ Similarity Ratio

It refers to simple possibilities that indicate how much the solutions are similar, i.e., how much the search is exploitive. This can be done by finding the final solutions generated by ants, and then apply some statistics tools such as the standard deviation and the variation coefficient. One example of these statistics is the standard deviation of the objective function of solutions constructed after each iteration, e.g., the length of the tours in TSP. The exploration amount will be high if the standard deviation is near to one, and low if near to zero.

Because of the dependency of the standard deviation on the scale of the problem, a better option is to use the variation coefficient. It is the quotient of the standard deviation and the average of the objective function of the generated solutions (Dorigo & Stützle, 2004). This measure is firstly utilized in the constraint satisfaction solver, as proposed by Solnon and Fenet (2005), in order to control the E&E balance.

This moves away from the previous tools in literature. Its aim is quantifying the exploration of the search space. Pellegrini, Favaretto, and Moretti (2009) have defined it as follows. Let a graph G= (N, A) represent a CO problem, where N is the set of nodes and A is the set of arcs, and |N| = n and |A| = a. Each solution *S* to this problem is denoted by $S_1 = \{x_1, x_2, \ldots, x_a\}$ and $S_2 = \{y_1, y_2, \ldots, y_a\}$. Each x_i represents the probability, denoted by P_i, of selecting this arc during the construction of solution *S*. The distance *D* between S_1 and S_2 is $D(S_1, S_2) = \sqrt{\sum_{i \in A}(d_i)^2}$ where $d_i = x_i - y_i$ if $x_i.y_i=0$ and $di=0$ otherwise.

Based on this definition, S_1 and S_2 can be clustered, if they are the closest solutions, using appropriate data clustering procedure. An agglomerative hierarchical procedure in data mining (Rajaraman & Ullman, 2012) is considered here. Iteratively, the maximum distance between the new cluster and the other solutions keep calculating until the stopping criteria. The stopping criteria stop when the distance between the two closest clusters is greater than a predefined threshold. Pellegrini et al. (2009) defined the quantity of exploration by the number of clusters built.

A new technique for measuring the effect of parameter variation based on this tool has been proposed by Favaretto et al. (2009). The study emphasized on using this tool due to its accuracy in indicating the actual behavior of the procedure. This technique may also serve as an indicator for defining parameter adaptation strategies. However, the technique restricts the parameter adaptation with offline approach. The online parameter adaptation may represent the first step in the direction of exploration and exploitation automation in ACO.

2.7 Reactive-based Parameters' Selection

The problem of parameter selection arises when an algorithm needs to select optimal values for its parameters from a relatively large search space of values. The balance between two opposing processes, namely, exploration and exploitation, has to be maintained in this selection. The parameters' selection differs not only from problem to problem, but also from region to region in the same search space. The situation is more complex and more sensitive with stochastic algorithms. The question is *what* is the methodology to handle this situation? According to Battiti et al. (2008), reactive search advocates the integration of subsymbolic machine learning techniques into search heuristics for solving complex optimization problems with an emphasis on opportunities for learning and self-tuning strategies (Figure 2.6).

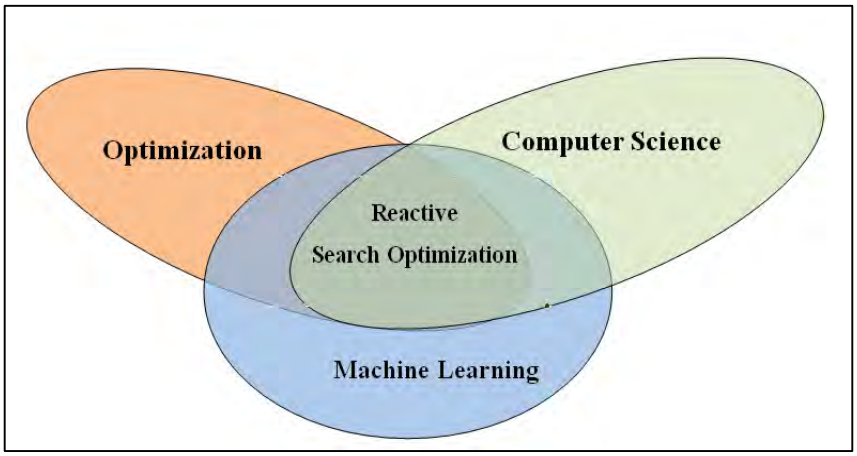


Figure 2.6. Reactive Search Optimization

The word reactive hints at a ready response to events during the search through an internal feedback loop and dynamic adaptation. In reactive search, the history of the search and the knowledge accumulated while moving in the configuration space is used for self-adaptation in an autonomic manner; the algorithm maintains the internal flexibility needed to address different situations during the search.

The next important question is *how* the exploration and exploitation in ACO are controlled based on reactive search? According to Lopez-Ibanez and Stützle (2014), ACO algorithms are just an example of problem-solving methods, which often have several parameters that allow the user to control the balance between exploration of new solutions and exploitation of the best solutions found. The study of the impact of various parameters on the behavior of ACO algorithms has been an important subject since the first articles of Colorni et al. (1991). However, a default parameter configuration is often used in practice, without taking into account differences in computational environment or termination criteria. This practice frequently leads to suboptimal results. Lopez-Ibanez and Stützle further stated, “One way to address this

issue is to not rely on default parameter settings, but instead to automatically configure the parameter settings." Pellegrini, Stützle, and Birattari (2010) justified using the automatic configuration by saying "to alleviate algorithm designers from the tedious and error-prone task of hands-on parameter adaptation". To date, proper management of control-parameter settings allowed the E&E balance to be achieved. However, which control-parameter setting to which yield the best result has remained to be answered. The parameters setting of metaheuristics are problem dependent and there is no optimal parameter setting which will work with every problem (Eiben, Michalewicz, Schoenauer, & Smith, 2007).

One possibility for automatic configuration task is following the offline parameter tuning approaches (Stützle & Lopez-Ibanez, 2013). Examples of offline approaches are F-Race (Birattari, Stützle, Paquete, & Varrentrapp, 2002), I/F-Race (Balaprakash, Birattari, & Stützle, 2007), CALIBRA (Adenso-Diiaz & Laguna, 2006), and ParamILS (Hutter & Leyton-brown, 2009). The main cost associated to offline algorithm configuration is the expensive use of resources in the priori experimental phase. Moreover, any search algorithm needs to be applicable for the vast domain of combinatorial problems, which entails that the amount of E&E needs to be changed dynamically with the optimizing process. Realizing this, the need for automating E&E balance, during the run, is imperative. It is foreseeable that better results can be achieved and faster convergence will occur when intelligent control is applied.

Another possibility is to change the parameter setting during the algorithm runtime, i.e. online approach, which is called parameter control. It is a radically different method. Pellegrini, Stützle, and Birattari (2010b) explained the difference: "In online tuning, no additional resource is necessary before actually tackling an instance and a high flexibility is achieved by adapting the configuration to an instance, depending on the specific phase of the search. This flexibility is paid for in terms of design complexity: the online method must be incorporated in the implementation of the optimization algorithm. Thus, a very good understanding of all different features of the algorithm is necessary for obtaining an effective online method. Hence, an online method must be designed for a specific algorithm, and it cannot directly be exploited on different ones."

Following the online approach, and according to Eiben et al. (2007) and Stützle, et al. (2012), there are four general questions need to be answered: i) what are the parameters that need to be changed during the search process?; ii) how will the change be made? (i.e., pre-scheduled, adaptive, search-adaptive and self-adaptive); iii) what is the evidence upon which the change is made? (e.g., quality of solutions, diversity of solutions, or the entropy of pheromone); and iv) what is the level of the change (i.e., ant level or colony level)?

Parameter control strategies modify the parameters during the run in different manners: pre-schedule, adaptive and self-adaptive. The first manner assumes that all CO problems are the same in their global characteristics, which is not true. The second manner adapts to the local characteristics of the regions of the search space through a feedback. It has the advantage of no augmentation in the complexity of the

problem. On the other hand, it suffers limitations: a complexity of implementation and presenting new hyper-parameters which also need to be tuned. The third manner has the advantage of tuning parameters "for free", where its implementation is simple and there are no hyper-parameters which need to be tuned. Besides increasing the complexity of the problem, it is linked to the structure of the algorithm.

2.7.1 Pre-Scheduled Approach

In this approach, the problem is observed from an offline perspective: static parameters are substituted by (deterministic or randomized) functions depending on the computational time or on the number of algorithm iterations. There is surprisingly little work on pre-scheduled parameter variation for ACO algorithms, where an algorithm tunes itself by scheduling its parameter variations with their iterations. Merkle and Middendorf (2001) were the first to study the effect of parameter variation during the run of the algorithm. They considered the ACO algorithm for the resource-constrained project scheduling problem. Their contribution was in decreasing the value of the parameter β linearly over the run of an algorithm due to concentrating the individual influence of β on the first iterations of the algorithm. Merkle, Middendorf, and Schmeck (2002) also considered the same problem and proposed to modify the parameter β, and the evaporation rate ρ. For β, they proposed a schedule as described before. For ρ, they proposed to start at a small value for increasing the initial diversification of the search space and to later set the evaporation rate to a high value for having an intensive search around the best solutions found by the algorithm. Meyer (2004) proposed another schedule for α (i.e. the pheromone influence parameter). The author exploited annealing scheduling in

simulated annealing method (Kirkpatrick, Gelatt, & Vecchi, 1983) to schedule the increase of the values of this parameter. Maur et al., (2010) examined two deterministic *MMAS* variants. In the first one, the parameter of the number of ants (m) starts with (1) and then increases by one every (10) iterations until the value becomes equal to the number of variables (n). In the second variant, the parameter of exploration/exploitation, denoted by (q_0), decreases starting from (0.99) until (0.0). Both variants showed good results in context of anytime behavior and quality of solutions. The same strategy is followed by Liu and Yang (2011) by considering more parameters. Alobaedy and Ku-Mahamud (2015) applied the strategic oscillation concept to control the exploration/exploitation parameter (q_0) after fixing the oscillation step size. The approach has been implemented on the colony level with positive results. The problem with such deterministic assignment of parameter values is that the number of iteration needed for convergence is unknown. In fact, devising proper values for parameters must be adapted based on the search progress.

2.7.2 Adaptive Approach

In adaptive approach, parameters are modified according to either the diversity of the pheromone trails or the quality of solutions. The searching behavior of the algorithm is considered. To determine this behavior, some measurement tools have been proposed, such as branching factor (Gambardella & Dorigo, 1995), entropy-based measure (Yancang & Wanqing, 2007) or exploration measure (Pellegrini et al., 2009).

The first adaptive approach in ACO was introduced by Yancang and Wanqing (2007). They varied the parameters α and β over time using the information entropy theory (i.e. the uncertainty of probability). They succeeded in adapting α and β values according to the searching algorithm. During the early stages of the search, the value of α is small to allow an extensive diversification of the search space; the value of α increases over time to improve the local search ability of the algorithm. They suggested the opposite adaptation for β. Zhiyong, Yong, Jianping, Youjia, and Xu (2008) proposed a variant of ACS that uses the cloud model proposed by Deren, Kaichang, and Deyi (2000) for electing the solution to be used to determine the amount of deposited pheromone. As in Yancang and Wanqing (2007), this work also exploits the entropy measurement tool to control parameter q_0 by decreasing it once much more by the pheromones concentrated on minority edges. Chusanapiputt, Nualhong, Jantarang, and Phoomvuthisarn (2006) proposed a method to solve the unit commitment problem using a variant of AS. Three of the algorithm's parameters are adapted using two modules for reducing the search space. The first module is for recording the infeasibility of some solutions to be avoided later, while the other is for recovering high-quality candidate path neighbors.

Zhaoquan, Han, Yong, and Xianheng (2009) and Zhifeng, Han, Yong, and Ruichu (2007) proposed a variant of ACS for TSP. They introduced a clear relation between the parameter ρ and the amount of pheromone associated with arcs. The main idea is that good ants have the higher pheromone. Amir, Badr, and Farag (2007) added a fuzzy logic controller module to the ACS algorithm for TSP for adapting the value of β and q_0. The adaptive approach uses two performance measures: i) the difference

between the optimal solution and the best one found; and ii) the variance among the solutions visited by a population of ants. Kov and Skrbek (2008) described a simple and effective approach to treat the colony as castes. Each caste of ants uses a different parameter setting. They indicated that a decreasing schedule for β can give a good performance. In addition, the number of ants *m* is adapted according to the improvement of the solution quality obtained by each caste. This approach successfully improves the convergence of the standard MMAS. Experimentally, the results are promising. However, they did not mention more details about their methodology.

In this approach, the adaptation rule can be applied when the measure being monitored hits a previously set threshold. For example, decreasing the diversity under a given value, the statistics and fuzzy rules can be considered as forms of evidence to apply the adaptation. Neyoy, Castillo, and Soria (2013, 2015) implemented fuzzy logic controller to adjust the pheromone concentrate parameter, denoted by (α). The rule of adaptation relied on *λ-branching factor* as an exploration indicator. Various fuzzy systems are proposed to control the diversity of solutions in order to maintain exploration and exploitation in *ACO* (Olivas, Valdez, & Castillo, 2015). Collings and Kim (2014) augmented the use of fuzzy controller for the adaptation-based, stagnation detection and control. Two fuzzy controllers were proposed by Liu et al. (2011) to adjust the parameters of the number of ants and the evaporation rate dynamically. The problem with such adaptation rules is that the user must determine the threshold values for triggering the rule activation. The users do

not have such intuition, whereas the algorithm itself has the ability to do that implicitly.

2.7.3 Search-Adaptive Approach

Search-Adaptive is a way to implicit-adapting the parameters of ACO algorithm, in which, the algorithm utilizes other search methods for adapting its parameters. Pilat and White (2002) used the GA method for adjusting some ACS parameters, namely, β, q0, and ξ. At each iteration, crossover and mutation operators are used to tune four of the ants' parameters before constructing solution. In the same way, Gaertner and Clark (2005) used ants to communicate with the environment and produce new solutions, while GA was chosen to exchange the new generations with the old ones. Ants are initialized with a random parameter setting within predefined ranges. The authors studied the parameter dependencies among β, q0 and ρ and conclude there was no statistically significant correlation to be found when the TSP problem is considered. Zhifeng, Ruichu, and Han (2006) did not present the relation between the parameters in their proposed self-adaptive approach for ACS in TSP. The parameters named β, q0 and ρ have been adapted using a Particle Swarm Optimization method (PSO) (Zhifeng et al., 2006), which selects the best values within a predefined range of parameters value. While in Weixin and Huanping (2007), Artificial Fish Swarm Algorithm (AFSA) has been used for the same purpose and focused on α, ρ and Q in a variant of ACS. As previous studies, the former method considered the ant's level, while the latter considered the colony level in the parameters variation. Garro et al. (2007) proposed another mechanism to adapt α, β and another specific parameter. They used crossover and mutation in GA to

evolve the generation of parameters to each kind of ant. In the crossover, the best explorer and worker ants' parameters are combined to generate a new offspring, and then one of the parameters is mutated. Anghinolfi, Boccalatte, Paolucci, and Vecchiola (2008) used local search to self-adapt two of variant ACS parameters named β and q_0, and then applied the enhanced method to solve single machine total weighted tardiness scheduling problem. The proposed method firstly increases and decreases the current parameter values by a fixed amount to produce the parameter space of the current setting. The neighbors of the current setting are locally searched by an ant that belongs to a different colony. Finally, the best iteration ant is allowed to exchange its setting with the old one. The multicolonies with multisetting paradigm is conducted in this approach, which is repeated in Melo, Pereira, and Costa's research (2010), which considered the following ACS parameters: α, β, ρ and q_0. The distinction between the two approaches is that the latter used mutation operator for exchanging the best setting with the worst one. The proposed mechanism has contributed a new measurement tool to indicate the disturbance of parameters, and then each parameter to be disturbed will be substituted by the best one in the best colony.

Following this approach requires encoding parameter settings so that the search-adaptive mechanisms can find the optimal adaptation. However, extending the solution size obviously increases the search space and makes the search process more time-consuming.

2.7.4 Self-Adaptive Approach

This is another way to implicit-adapting for ACO parameters, in which, the algorithm utilizes itself instead of using other search methods for adapting its parameters. The first work was introduced by Randall (as cited by Gaertner & Clark, 2005 and Stützle et al., 2010). He suggested evolving parameters based on an extra pheromone matrix which are maintained solely for this purpose known by parameters matrix. This mechanism is tested by adapting the parameters β, q0, ρ, and ξ for ACS applied to TSP and the quadratic assignment problem. The comparison of the results to the default parameter settings is somehow inconclusive. As in Randall's study, Forster, Bickel, Hardung, and Gabriella (2007) applied a parameter matrix to adapt parameters, where each column represents a parameter and each row represents a different value to that parameter. Each ant has to construct a tour of its own parameter setting. In the two previous approaches, there are no dependencies between parameters. In contrast, Martens, Backer, Haesen, Vanthienen, and Snoeck (2007) proposed a self-adaptive approach based on the interdependent relation between α and β. In his ACO method, which is conducted as a classification task, each parameter value varied through a new vertex group in the construction graph. Khichane, Albert, and Solnon (2009) proposed two methods for tuning parameters α and β during the run of his ACO algorithm to solve constraint satisfaction problem. The two methods defined parameter setting for the colony and not for each ant. Similar to the work by Forster et al. (2007), Gaertner and Clark (2005), and Stützle et al. (2010), the method did not consider the interrelation between the parameters. However, the methods in Khichane et al. (2009) focused on learning the parameters

during the construction solution phase in two ways: a new parameter setting in each construction step, or a new parameter setting for all construction steps.

According to Battiti and Protasi (2001), these approaches adapt typically very few (often only one) key parameters of an algorithm and require substantial insight into the algorithm's behavior for their development. The challenge, however, is that the E&E balance in ACO is implicit, and as such, controlling it directly is difficult. Therefore, gaining a better understanding of the E&E balance requires knowing how to measure it.

2.8 Discussion on Reactive-based ACO

The rationale works in terms of E&E in ACO are the combination of pheromone management with local search procedures, auxiliary memories or local heuristic information and parameter adaptation (Battiti et al., 2008). This section discusses these categories together in order to reach a unified proposal for the exploration and exploitation problem.

For the pheromone management, several strategies have been reviewed in this chapter. From the E&E perspective, the difference of E&E behavior is based on the amount of exploration promoted by these strategies. The findings of the various published research papers on AS extensions indicate that the best performing variants are MMAS and ACS. The aggressive exploitation in ACS produces good solutions for very short-term runs. Conversely, MMAS starts with a long exploration so that its early quality solutions are poor. Nevertheless, the final solution quality of

MMAS is the best among other ACO algorithms especially for long-term runs. The relative good performances of ACS in short-runs and MMAS in long runs are interpreted by the difference in their exploration/exploitation behavior. Figure 2.7 provides a clear picture about the behavior's difference among some well-known ACO algorithms.

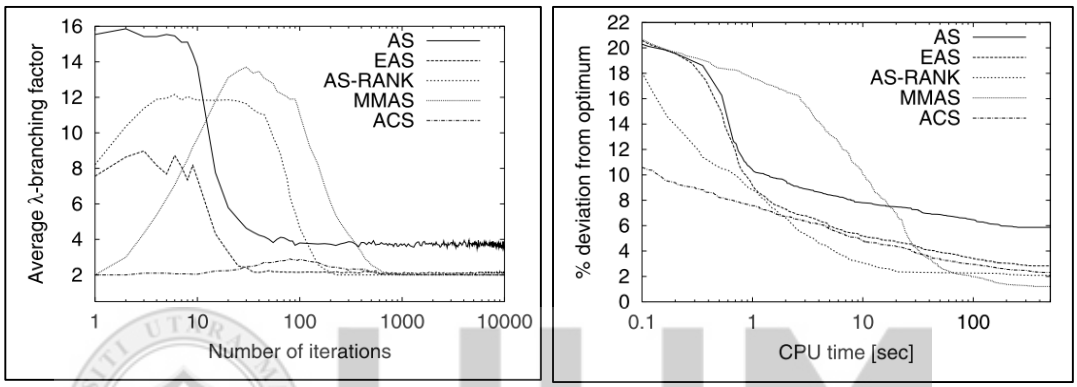


Figure 2.7. Variance of Exploration and Exploitation Behaviour in ACO

These experiments have conducted by Dorigo and Stützle (2004) on the symmetric TSPLIB instance *kroA100* where the percentage deviation from optimum refers to the quality of solutions and the average lambda branching factor refers to the exploration behavior. It can be concluded that although the search strategy of MMAS enables finding high quality solutions it tends to be over-explorative comparing with other algorithms. To ensure this point of view, Table 2.2 provides a comprehensive conceptual comparison between E&E search strategies in ACO. This comparison can be generalized for other ACO algorithms such as ANTS and Beam-ACO by adding one point for each ACO variant based on its specific search strategy. For example, the best-worst and online-offline strategies will score one points for ANTS as they are part of its structure while hybridizing with local search will score one point for

all ACO algorithms as its generic-exploitation purpose (Gambardella et al., 2012; Perez-Caceres, Lopez-Ibanez, & Stützle, 2014).

Table 2.2

Amount of Exploration and Exploitation in ACO Algorithms

Exploration and Exploitation in ACO Algorithms	**AS**	**EAS**	**AS-rank**	**MMAS**	**ACS**	**BWAS**	**PACO**	**AntNet**	**MACO**
Quality-dependent	√	√	√	√	√	√	√	-	√
Quality-independent	-	-	-	-	-	-	-	√	-
Elitist	-	√	√	√	√	-	-	-	-
Rank-based	-	-	√	-	-	-	-	-	-
Trail learning	-	-	-	√	√	-	-	-	-
Online-offline update	√	-	-	-	√	-	-	-	-
Best-worst	-	-	-	-	-	√	-	-	-
Bounding	-	-	-	√	-	-	-	-	-
Smoothing	-	-	-	√	-	-	-	-	-
Restarting	-	-	-	√	-	-	-	-	-
Colony-level interaction	-	-	-	-	-	-	-	-	√
Population-based	-	-	-	-	-	-	√	-	-
Hybridizing	√	√	√	√	√	√	√	√	√
Score points	3	3	4	**7**	5	3	3	2	3

The total score of MMAS was seven (7) comparing with others. For this end, the MMAS outperforms others experimentally and conceptually. Consequently, MMAS has been selected in the present study to be the base for more advanced exploration and exploitation components.

For more advanced improvements on the top of MMAS other than the hybridization with local search, a population memory vector, denoted by (*P*), for deriving new pheromone management models is added (Oliveira, Stützle, Roli, & Dorigo, 2011). The amount of pheromone added/dropped relies on the size of the memory *P* which is denoted by |P|. This approach contributes in faster pheromone updates (Oliveira et al., 2011) and motivates the invention of more advanced structural features (Lin & Middendorf, 2013). However, the local search may become time-consuming when

the running time is tight or the computation of solution evaluation is high. Additional weakness in that the population memory vector is not able to transfer neighborhood structures that are formed either by construction solution procedure or by local search procedure from current iteration to future iterations.

Another recurring theme in terms of exploration and exploitation is the harnessing of pre-heuristic information for the selection of solutions' components. This E&E component plays a profound role in improving the internal behaviour of any ACO variant other than MMAS (Ku-Mahamud & Alobaedy, 2013). However, its priori availability is not guaranteed in problem-solving which may restrict its application and increase its limitation. Another shortcoming is that the formulation of heuristic functions is difficult and required deep knowledge about the CO problem under tackle which indeed will impose additional burden on the algorithm designer. These gaps entail building new kinds of internal heuristics in on-the-fly fashion which induce adapting reactive search characteristics (Battiti et al., 2008) based on the concept of "learning while optimizing". Toward building effective ACO-based reactive search methods, Solnon (2010) stressed on learning more ACO parameters and using exploration indicators, such as the similarity ratio betweeen the solutions of current population, in parameter adaptation. A schematic description about the distribution of reactive characteristics in the literature is provided in Tables 2.3 and 2.4.

Among several exploration measures reviewed in this study, there are three recurring exploration indicators used for parameter adaptation. Those are the entropy, λ-branching factor and similarity ratio.

Table 2.3

Schematic Description of the Literature on ACO-based Reactive Search

Authors	Number of Parameters	Exploration Indicator	Reactive Characteristic	ACO Model	CO Problem
Merkle et al.(2002)	m, q_0	deterministic	LAC, LHP, SI, ASI	AS	scheduling
Meyer (2004)	m	deterministic	LAC, LHP, SI, ASI	AS	TSP
Maur et al.(2010)	α , β	deterministic	LAC, LHP, SI, ASI	MMAS	TSP, QAP
Liu and Yang (2011)	q_0	deterministic	LAC, LHP, SI, ASI	MMAS	VRP
Alobaedy and Ku-Mahamud (2015)	q_0	deterministic	LAC, LHP, SI, ASI	ACS	scheduling
Yancang and Wanqing (2007)	α , β	entropy	AGC, ALC, ASI	AS	TSP
Zhiyong et al.(2008)	q_0	entropy	AGC, ALC, ASI	variant of ACS	TSP
Chusanapiputt et al.(2006)	specific parameters	deterministic	AGC, ALC, LAC, ASI	variant of AS	industry
Zhaoquan et al. (2009)	ρ	deterministic	AGC, ALC, LAC, ASI	variant of ACS	TSP
Zhifeng et al.(2007)	β , q_0	deterministic	AGC, ALC, LAC, ASI	variant of ACS	TSP
Amir et al.(2007)	β , q_0	entropy	AGC, ALC, ASI	ACS	TSP
Kov and Skrbek (2008)	β, m	deterministic	AGC, ALC, LAC, ASI	MMAS	TSP
Neyoy et al.(2013)	α	branching factor	AGC, ALC, LAC, ASI	RAS	TSP
Collings and Kim (2014)	α , β, ρ	branching factor	AGC, ALC, LAC, ASI	RAS	TSP
Liu et al. (2011)	ρ ,m	deterministic	AGC, ALC, LAC, ASI	AS	Feature selection
Olivas et al. (2015)	α , ρ	Similarity ratio	AGC, ALC, LAC, ASI	RAS	TSP
Gaertner and Clark (2005)	β, q0 , ρ	Relative	AGC, LHP, SI, GM	ACS	TSP
Weixin and Huanping (2007)	specific parameters	Relative	AGC, LHP, SI, GM	variant of ACS	TSP
Zhifeng et al.(2006)	β, q0 , ρ	Relative	AGC, LHP, SI, GM	ACS	TSP
Garro et al. (2007)	specific parameters	Relative	AGC, LHP, SI, GM	variant of AS	path-planning
Anghinolfi et al.(2008)	β , q_0,	Relative	AGC, LHP, SI, GM	variant of ACS	scheduling
Melo et al. (2010)	α, β, ρ , q_0	Relative	AGC, LHP, SI, GM	MACO	Node placement
Randall (2004)	β, q0, ρ, ξ	Relative	AGC, LHP, SI, GM	ACS	TSP
Martens et al.(2007) and Förster et al. (2007)	specific parameters	Relative	AGC, LHP, SI, GM	variant of MMAS	Feature selection
Khichane et al. (2009)	α , β	Relative	AGC, LHP, SI, GM	variant of MMAS	Car sequencing

Table 2.4

Abbreviations of the Reactive Characteristics

Reactive Characteristic	Abbreviation
Adapted with global characteristics	AGC
Adapted with local characteristics	ALC
Less augmented complexity	LAC
Less hyper parameters	LHP
Simple implementation	SI
Algorithm structure independent	ASI
Follow a general methodology	GM

For applications of entropy (Yancang & Wanqing, 2007; Zhiyong et al., 2008; Amir et al., 2007), they associated with high augmented complexity as it complicates the calculations inside the colony. For the applications of λ-branching factor (Collings & Kim, 2014; Neyoy et al., 2013, 2015), the disadvantage is the dependency of branching factor on the value of its lambda parameter. Moreover, it is ineffective for analyzing the exploration behavior for CO problems other than TSP.

For the similarity ratio, Olives et al. (2015) utilize the concept of fuzzy logic to schedule the value of parameters which can be considered as implicit deterministic approach. The problem with deterministic way is that the parameter adaptation is pre-scheduled according to a function of variation in the number of iterations.

In exloration measurement literature, Pellegrini and Favaretto (2012) quantified the exploration as the number of clusters of solution visited. So far, this indicator, namely exploration measure has not applied in parameter adaptation. This indicator utilizes machine learning procedures to provide online heuristic information during the search. From reactive search point of view, the process of learning parameters, either offline (Lopez-Ibanez & Stützle, 2011; Pellegrini, Stützle, & Birattari, 2010a)

or online (Neyoy et al., 2013, 2015; Stützle et al., 2012), it must be coupled with machine learning procedures.

Following this methodology, the integration of exploration measure with reactive-based parameter adaptation methods may improve their performance. However, the exploration measure has a robust problem in its functionality to find the similarity between two clusters which can be concluded when the distance between them is greater than a predefined threshold ϵ_x where $x\%$ of their arcs does not exist in the cluster. In TSP, $\epsilon_x = 7.8$ in MMAS without local search, $\epsilon_x = 17.5$ in MMAS with 2-opt local search and $\epsilon_x = 35.8$ with 3-opt local search. The situation changes with the change of circumstances. This definition needs to be reconsidered in terms of robustness against the abovementioned situation which leads to an unstable measurement, especially when flat fitness landscape needs to be analysed.

For the applications of relative indication, the adaptation relies on the improvement in quality of solutions produced by outer optimization scheme, i.e. the ACO algorithm itself as exemplified in the works of Anghinolfi et al. (2008), Förster et al. (2007), Khichane et al. (2009), Martens et al. (2007) and Randall (2004). Among them Khichane et al. (2009) and Randall (2004) have built their adaptation on the top of MMAS and ACS respectively. Following the implementation of these two methods, a critical analysis of parameter adaptation methods in ACO has been conducted in a comparison with MMAS by Pellegrini et al. (2012) showing the superiority of MMAS in standard benchmark CO problems such as TSP and QAP. Although Randall (2004)'s method succeeded in improving the performance of the standard ACS it does not provided any insights about MMAS. These empirical

results confirm that MMAS is eligible to be a base paradigm for building success reactive parameters' adaptation methods. This selection not only justified by the E&E strategies included in MMAS (see Table 2.2), but because of the fact that MMAS is very sensitive for adapting its parameters. From parameter adaptation search point of view, existing self-adaptive approaches in ACO are a good candidate if they adhere well to reactive methodology by utilizing more robust exploration indication in parameter learning.

2.9 Summary

The exploration versus exploitation dilemma is resident in ACO. Due to its importance, several strategies have been proposed to address it. The differences among ACO algorithms can be identified from their way of addressing this problem. For the self-contained background, AS algorithm, as the first proposed ACO algorithm, and MMAS algorithm, as one of the best-performing variants of AS, have been highlighted in this chapter. A conceptual comparison with other AS variants has been provided as well. Reactive framework is designed to achieve a proper E&E balance. The main aspects of reactive search to be applied in ACO are the memory model, the exploration measure, i.e. the feedback and the on-the-fly scheme for parameters' selection. By the feedback scheme, a track of an exploitative search using local search is performed in order to promote exploration only as needed using restart. The feedback is the core of adaptive parameters' selection methods. The abovementioned aspects have been detailed separately in their perspective sections. Finally, unified insight and interrelation among the various E&E aspects in related

studies have been provided at the end of this chapter. In Chapter 3, the methodology used to undertake the research is presented.

CHAPTER THREE
RESEARCH FRAMEWORK AND METHODOLOGY

3.1 Introduction

This chapter presents the framework and methodology of this research. It starts with Section 3.2 that depicts the research framework and the methods used to achieve the objectives of the research. Based on the research objectives, the proposed methods are presented in Section 3.3, which briefly explains the roles of each method and the experimental design and the selected benchmarks for evaluating each method separately and the whole approach. Finally, this chapter is summarized in Section 3.4.

3.2 The Research Framework

The high-level focus of this research is to propose a reactive approach that addresses the exploration and exploitation in ACO. The approach comprises of four steps; memory model development, exploration measurement enhancement, adaptive parameters' selection and evaluation as in Figure 3.1. The first step to maintain this balance is changing the principle of "later aggressive exploitation" in MMAS memory model to "minimal exploration only if needed" and the principle of "long initial exploration first" in the same model to "exploitation is first". This goal is achieved by twofold processes. Firstly, developing reactive heuristics as local heuristics in the transition probabilistic rule of construction solution function where ants' experience can be transferred over restarts. Secondly, developing recursive

local search technique based on the scheme of population-based memory where previous population is archived and then improved by local search.

The second step of exploration and exploitation balance concerns the feedback from the current search process in terms of population distribution. A machine learning indicator has been developed to characterize the over-exploitation state in general, not restricted to MMAS model only. For the state, it triggers the restart of search with the aid of local heuristics that recorded in terms of reactive heuristics. As opposed to this state, over-exploration is rare due to the role of recursive local search.

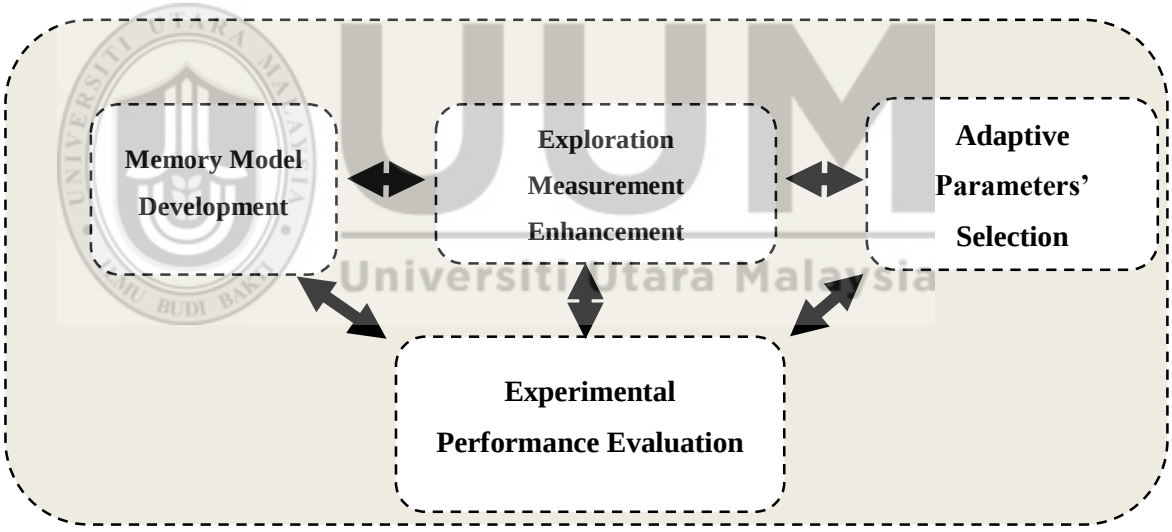


Figure 3.1. High-level Research Framework

In step three, the adaptive parameters' selection provides automatic control of algorithmic parameters while solving the problem to improve the search efficiency. In order to do so, one needs to define the reward assignment scheme which rewards parameters based on the feedback from current search process. Different reward assignment schemes have been proposed to calculate the amount of reward to given

to the promising parameters. These are based on quality of solutions and/or diversity of solutions.

The fourth step describes the experimental approach conducted to validate each of the above mentioned steps developed together or separately. In all existing metaheuristics, the experimental research was the main guide to design and develop any novel algorithm (Barr, Golden, Kelly, Resende, & Stewart, 1995). This methodology is useful for characterizing and understanding the complex behavior of the metaheuristic algorithms (Talbi, 2009). ACO metaheuristic is a history of experimental research (Dorigo & Stützle, 2004). This research follows the same methodology of designing the most successful ACO algorithms. The ultimate goal of parameter optimization and diversity strategies in this research is to improve algorithm performance, which results in better convergence behavior. Figure 3.2 extends the aforementioned steps in detail and gives a conceptual view about the proposed approach.

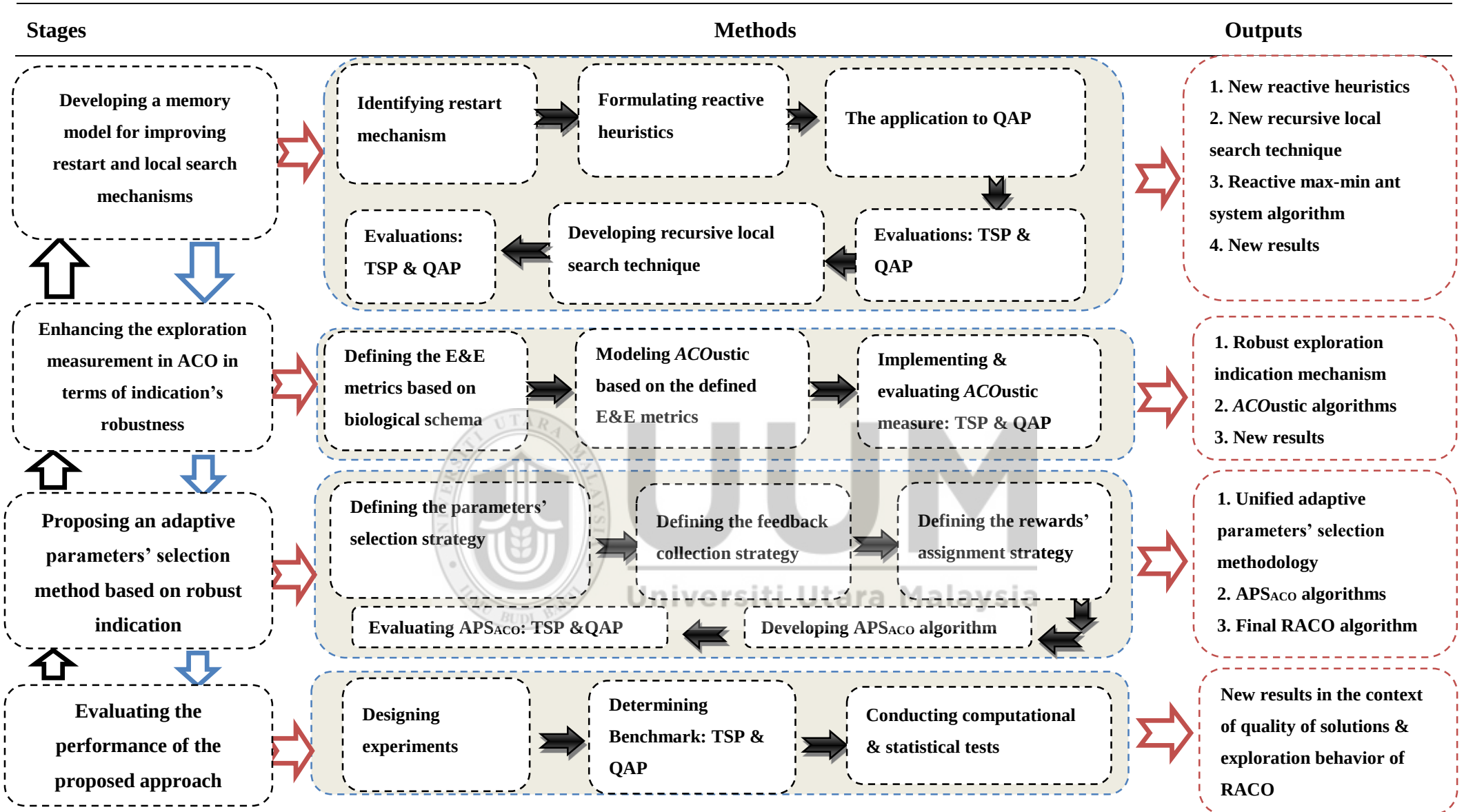


Figure 3.2. Low-level Research Framework

3.3 Research Methods

This section presents the proposed methods and draws the roadmap for understanding the proposed approach. The memory development method is presented in Subsection 3.3.1. The exploration measurement and the online parameter's selection methods are highlighted in Subsection 3.3.2 and Subsection 3.3.3 respectively.

3.3.1 Developing the Memory Model

An auxiliary memorizing feature is added to control the probabilistic distribution after restart and to concentrate the search around the neighborhood of solutions produced by local searches. The memory model development includes two algorithmic components: reactive heuristics and recursive local search. The first component is defined before the search starts: RH =def [RH_0]. The size of RH is equal to the pheromone model size. In the evaporation update, the arcs with a small pheromone amount are recorded in the model: $rh_{i,j}$= {1, 0}. RH is reactivated when the stagnation occurs, i.e. at the point of restart. Then, it will be considered as a new input to the transition state rule: *TourConstructionSolution (T, C, RH)*, where T is the pheromone information, and C is the heuristic information. Therefore, the ants will select the insignificant arcs that are neglected before to increase the exploration behavior.

The recursive local search technique, the second algorithmic component, is designed to solve the premature exploitation where the current neighborhood structure is not transfered to next iterations using current local search procedures. A population

vector called *P* is designed to track the best-so-far solutions, the best-iteration solutions, and the old-best solutions. The first two solutions are the output of current local search, while the third solution is just a dropped solution from the vector *P*. The old-best solution is added again and again to the population if and only if the best-so far solution is not improved by local search. More details about this phase are discussed in Chapter Four.

3.3.2 Enhancing the Exploration Measurement

In this phase, the current exploration measurement is enhanced. Three criteria are the main proxies for this measurement: the variation in quality of solutions (ΔOG), the variation in diversity of solutions (ΔR) and the combination of both of them. The ΔR and ΔOG are analogies to exploration and exploitation respectively. Among several exploration measures in ACO literature, a few of them are compatible with the reactive search framework, while the rest are suffering a robust problem where the distance matrices that determine the diversity of solutions are in a different magnitude. Nature-inspired solutions to the problem are proposed. Both the biological and computational schemes are detailed in the first part of Chapter Five.

3.3.3 Proposing Adaptive Parameters' Selection Method

Parameter adaptation is a high level control of exploration and exploitation balance. In this phase, three methods of adaptive parameters' selection are proposed based on the feedback from the optimization process, i.e. the evidence that the current parameter values have succeeded in improving the quality of solutions. Here, the three exploration measures will be used as proxies to indicate the improvement.

Subsequently, good parameter values need to be awarded in an online reinforcement learning fashion. Three reward assignment strategies can be found in the second part of Chapter Five.

3.4 Evaluation of the Proposed Approach

In this stage, the algorithmic components of the proposed approach and the overall algorithm are evaluated. The evaluation links to all the above-listed stages and gives more flexibility to revise any stage for better performance. The performance evaluation of metaheuristics is very complex, and hence, Talbi (2009) listed three steps to conduct it in a fair manner. These are experimental design, measurement and reporting as shown in Figure 3.2.

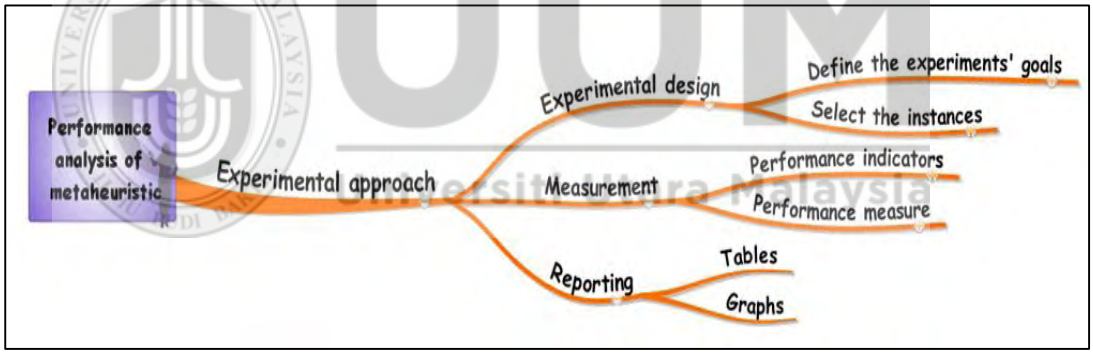


Figure 3.2. Performance Evaluation of Metaheuristic Algorithm

As the experimental methodology has matured in the metaheuristic area, there have been increasing demands for a more careful evaluation using a good experimental design. To achieve this goal, two methodological aspects need to be conducted, which are defining the goals of the experiments and selecting the instances. The main goal of designing the experiments is to evaluate the quality of solutions and the robustness of the proposed methods. There are three kinds of solutions, namely the

optimal solutions, the iteration-best solutions and the bestso-far solutions. In terms of robustness, the design of experiments can show both the robustness of CO problem instances and the robustness of the algorithms' parameters.

According to Johnson (2001), achieving meaningful and publishable results is harder than the coding of an algorithm in the benchmarking analysis. Hence, a lot of efforts have to be channeled in assessing the results and investigating the behavior of the new algorithmic components. For the TSP coding, the implementation of Stützle (2004), i.e. ACOTSP.V1.3 software, is used. The c code has been released in the public domain and is available for free download on http://www.aco-metaheuristic.org/aco-code/. For the QAP coding, the implementation of Taillard (2010), i.e. FANTQAP software, is used. The c code is available for free download on http://mistic.heig-vd.ch/taillard/codes.dir/fant_qap.c. Because of the similarity in the structures of TSP and QAP, the ACOTSP.V1.3 software has been extended to fit the QAP modeling for other ACO variants based on the implementation of Taillard. The validation for solving the QAP problem called *tai10b.dat* with the value five (5) for parameter R and the value hundred (100) for number of iterations gives the similar output as of Taillard (2010).

The performance of any proposed algorithm can be determined statistically if it is compared to other algorithms when solving the same problem instances (Lafayette, 2001). Therefore, and after defining the experiment's goals, the selection of instances has been selected carefully. The test-beds will be two of the combinatorial problems, which are traveling salesman problem (TSP) (Lawler, Lenstra, Kan, &

Shmoys, 1985) and quadratic assignment problem (QAP) (Lawler, 1963). The following subsections provide detailed descriptions about the problems and how they can be modelled in ant colony optimization.

3.4.1 The Traveling Salesman Problem

The importance of TSP arises because of the extensive studies and the high recommendations by computer scientists to be used in the evaluation of new optimization algorithms. This problem has been proven as an NP-hard problem. It can be described as follows. An agent has to visit *N* nodes exactly once and return to the starting node with minimum cost, i.e. the shortest distance or the lowest visiting time. A cost matrix $C = [c_{ij}]$ is searched to find a permutation $\pi : \{0, \dots, N-1\} \rightarrow \{0, \dots, N-1\}$, where c_{ij} represents the cost of visiting node (j) from node (i). The goal is to minimize an objective function denoted by $f(\pi, C)$ as follows.

$$f(\pi, C) = \sum_{i=0}^{N-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(N)}, c_{\pi(1)}) \quad (3.1)$$

where $\pi(i)$ represents the i^{th} node in permutation π, d is the distance between nodes and $c_{ij} = c_{ji}$ $\forall i, j$ and the position of city (i) can be determined using the values of x-axes, y-axes, i.e. x_i and y_i respectively, Hence, the cost matrix is calculated as follows.

$$c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3.2)$$

The dataset instances are taken from TSPLIB (Reinelt, 1991) benchmark library. TSP instances used in the experiments are classified according to their sizes (n):

small size where n=50-100 (such as eil51, berlin52, st70, eil76, pr76, gr96, rat99, kroA100, kroB100, kroC100, kroE100 and rd100), medium size where n=100-800 (such as d198, lin318, pcb442, att532 and rat783), and large size where n> 800 (such as pcb1173, d1291 and fl1577). Figure 3.3 simplifies one type of TSP instances extracted from TSPLIB (see Appendix A for more detailed discretion).

```
NAME: burma14
TYPE: TSP
COMMENT: 14-Staedte in Burma (Zaw Win)
DIMENSION: 14
EDGE_WEIGHT_TYPE: GEO
EDGE_WEIGHT_FORMAT: FUNCTION
DISPLAY_DATA_TYPE: COORD_DISPLAY
NODE_COORD_SECTION
1     16.47    96.10
2     16.47    94.44
3     20.09    92.54
4     22.39    93.37
5     25.23    97.24
6     22.00    96.05
7     20.47    97.02
8     17.20    96.29
9     16.30    97.38
10    14.05    98.12
11    16.53    97.38
12    21.52    95.59
13    19.41    97.13
14    20.09    94.55
EOF
```


Figure 3.3. Sample Structure of TSPLIB File

Selecting different structures of instances gives more understanding to the behavior of the proposed algorithmic components when tackling the TSP problem. In all instances, the n nodes represent specific locations in specific cities, e.g. Burma. The first five lines include some information about the problem being tackled, such as the data type, whether Euclidean, geographical or other types. The TYPE keyword specifies the type of data, e.g. symmetric, asymmetric or a collection of tours. The

keyword DIMENSION is the number of nodes for the TSP instances. The keyword EDGE_WEIGHT_TYPE specifies how the edge weight is defined, e.g. the keyword EUC_2D is the Euclidean distance in the plane, while the keyword GEO is geographical distance. The keyword NODE_COORD_ SECTION starts the node coordinates section. Each line is made of the node identifier, x and y coordinates. The node identifier is a unique integer ≥ 1. The statistics about several TSP instances are summarized in Table 3.1.

Table 3.1

Description of Some TSP Instances

File name	**Location**
att532.tsp	Padberg/Rinaldi
berlin52.tsp	Berlin (Germany)
bier127.tsp	Juenger/Reinelt
burma14.tsp	Burma (Myanmar)
d198.tsp	Reinelt
eil51.tsp	Christofides/Eilon
fl1577.tsp	Reinelt
gil262.tsp	Gillet/Johnson
gr96.tsp	Europe
kroA100.tsp	Random
lin318.tsp	Lin/Kernighan
pcb442.tsp	Groetschel/Juenger/Reinelt
pr299.tsp	Padberg/Rinaldi
rat783.tsp	Pulleyblank
rd100.tsp	Reinelt
st70.tsp	Smith/Thompson

3.4.2 The Quadratic Assignment Problem

QAP is the hardest NP-hard problem. It has an important theoretical value in the study of the behavior of high performance algorithms. It can be described as a problem of assigning economic facilities to locations while minimizing costs as in Figure 3.4. A set of facilities (n) needs to be assigned to a set of locations (n) with given distances between the locations and given flows between the facilities.

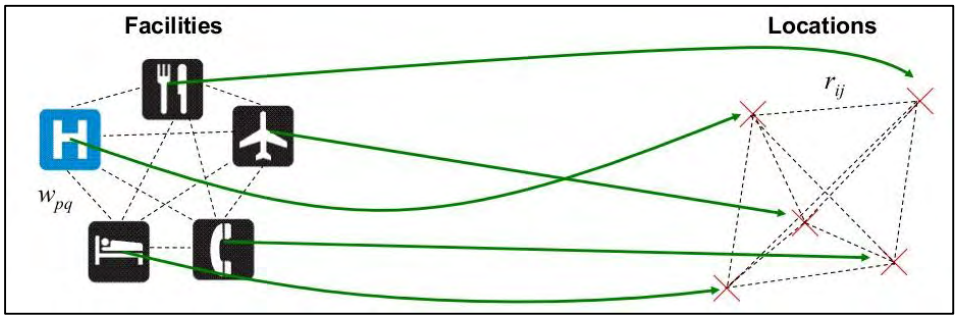


Figure 3.4. High Level Description of QAP

The flows and locations are two $n \times n$ matrices dented by W and R respectively, where w_{pq} is the flow between facility p and q and r_{ij} is the distance between location *i* and *j*. The objective is to place the facilities on locations in such a way that the sum of the product between flows and distances is minimal. The objective function denoted as $f(\emptyset)$ can be formulated as follows (Stützle, 1999).

$$f(\emptyset) = \sum_{p=1}^{n}\sum_{q=1}^{n} w_{pq} r_{\emptyset p \emptyset q} \tag{3.3}$$

Let the flows and distances matrices complete undirected graphs whose edges will be valued after the assignment as designated in Figure 3.5. A QAPLIB instance file contains the size of matrices and the seed for random generated instances followed by facilities and locations matrices as in Figure 3.6, which describes the QAPLIB file, namely *tai10b.qap*.

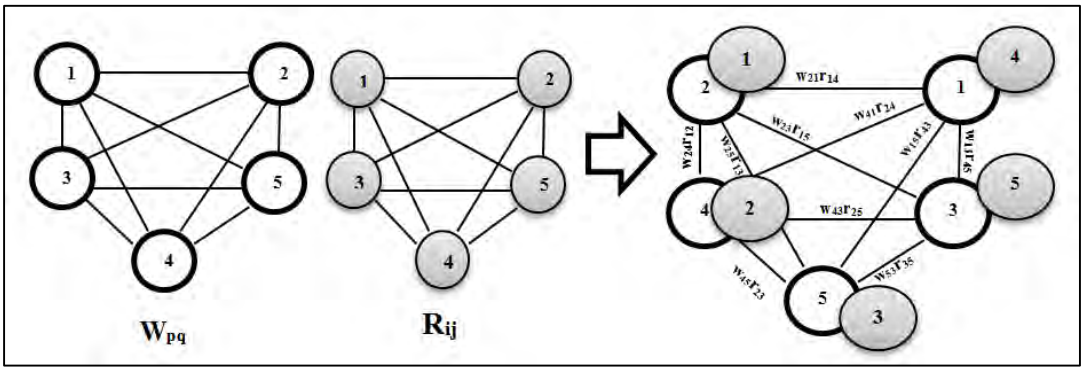


Figure 3.5. A Graph Model for QAP Relaxation

```
10
1183760

 0   19   78   60   79   45   65   37  103   34
19    0   65   45   76   63   79   22  109   19
78   65    0   21   44  113  104   72   97   73
60   45   21    0   53   99   97   50  102   51
79   76   44   53    0   98   74   93   54   92
45   63  113   99   98    0   42   81   97   78
65   79  104   97   74   42    0  100   57   98
37   22   72   50   93   81  100    0  130    3
103 109   97  102   54   97   57  130    0  128
34   19   73   51   92   78   98    3  128    0

 0    1    0    2    0    0   15    0    0  172
171   0    0   14    0   61   30    0  886   45
 43   0    0 2106    1    0    0    0    0    0
361   0    0    0    0    0    0    0    0    0
  2 123    1    0    0   49   18    0  335 2417
1096   0    0    0    0    0  952    5    0    0
 207 3703   27    0    4    0    0    0  202    0
  16  58    0    0    0    0  546    0   42 1213
   0   0    0   53    0  546    7 2649    0   86
   0   0 6707    1   12 7124    1    0    1    0
```

Figure 3.6. Sample Structure of QAPLIB File

The QAP instances with their feasible solutions are listed in alphabetical order by the names of their authors in Appendix B. The instances are taken from QAPLIB benchmark library (Burkard, Cela, Karisch, & Rendl, 1997). The QAP instances can be classified according to their structure into real-life instances (such as *bur26a, bur26b, bur26c, bur26d, chr25a, els19 and kra30b*), real-life-like instances (such as *tai20b, tai25b, tai30b, tai35b, tai40b and tai80b)*, and random-generated instances (such as *nug30, ste36b, tai30a, tai40a, tai50a, tai60a, tai80a and tai100a*). Besides

the theoretical importance of QAP, it can be derived practically from various engineering designs, e.g. integrated circuit wiring, job scheduling, the typewriter keyboard design, and hospital layout.

3.4.3 Benchmark Methods

The main goal of the thesis is to build exploration and exploitation components that improve ACO-based reactive search, which result in better algorithm performance. To determine if the goal is achieved, the components introduced in this thesis are compared with distinguished benchmark methods. For each of the three contributions, the achieved results are compared against the benchmark methods which correspond to that contribution. Benchmark methods are described in the experimental design of each contribution chapter. Thereafter, the three contributed components are combined in a unified algorithm to evaluate the overall performance of the proposed reactive approach. The results of the unified algorithm are compared with several metaheuristics approaches for solving TSP and QAP. The description about the algorithms with which the proposed approach is compared is presented in the experimental design of Chapter Six.

3.4.4 Comparative Measures

As a stochastic method, ACO is not expected to give repeated or exact results, but approximate results. In order to measure and compare the performance of two or more methods, an accurate evaluation has to be executed. The computational performance of algorithms can be assessed by *CPU Time Measure (CTM)* to evaluate the speed of convergence. However, CTM is inconsistent with the principal

of accuracy (Moret, 2001). Eiben and Jelasity (2002) explained how the results may vary based on the programmer's experience, the compiler, and the operating system. Johnson (2001) provides the *Success Rate Measure (SRM)*, which is the suitable method to evaluate the convergence behavior after applying the new methods. SRM is the percentage of runs that terminate with success (i.e., finding the optimal solution). Therefore, it has been used in the first experiments in Chapter Four.

Another suitable way to measure the computational performance is to use the *Quality Solution Measure (QSM)* (Aleti, 2012; Hooker, 1995). It is the mean of the best-so-far solutions over the number of allowed iterations. This way of measurement is used frequently in the experiments conducted in Chapters Four, Five and Six. To restrict the randomness effect, each experiment runs 10-30 times. The cost results are reported as the relative percentage deviation (RPD) from the best known solution cost. This is calculated as follows (Lopez-Ibanez & Stützle, 2014).

$$((\text{the result cost} - \text{the best known cost}) / \text{the best known cost}) \times 100 \quad (3.1)$$

Note that "*min*", "*med*" and "*max*" represent the minimum, median and maximum RPD respectively.

Non-parametric statistical tests, such as Wilcoxon signed-rank (Wilcoxon, 1945) and Chi-square (Battiti & Birattari, 2013), are used to confirm the significance of obtained results at the 95% confidence level. The expected results produced from the previous steps will be interpreted using the predefined goals of experiments. The

results will be reported in a graphical way so that the means is distinct and the performance is distinguishable.

3.5 Summary

This research aims to develop a metaheuristic algorithm that is able to manage the exploration versus exploitation dilemma in ACO. This entails a methodology that can guide the understanding of the complex exploration/exploitation behavior of the ACO algorithm. This research follows the experimental methodology which is used in the development of the most successful metaheuristics. It is impossible to omit the experimental approach from the metaheuristic algorithm development. To complete this aim satisfactorily, the three principles that should govern the experimental research methodology: generalizability, performance measures, and reproducibility have been covered. The experimental approach used to evaluate the performance of metaheuristics in a fair manner has been discussed and considered as a guide in the development of the proposed methods. The first proposed method, namely memory model development, is detailed in Chapter 4.

CHAPTER FOUR
MEMORY MODEL DEVELOPEMENT AND ITS APPLICATIONS

4.1 Introduction

This chapter describes the proposed memory model to improve the restart and local search strategies in max-min ant system, the prominent ACO variant. The reactive heuristics and recursive local search technique are the two algorithmic components that are added to MMAS. A new MMAS variant is proposed based on MMAS, which is the reactive max-min ant system (RMMAS), and an exploitation mechanism called recursive local search (RLS). Section 4.2 shows the process of developing the proposed memory model. It includes the formulation of new reactive heuristics, their application to QAP, the experimental design for evaluating RMMAS algorithm, and the results of evaluation then followed by the development and the experimental design for evaluating of the RLS technique and the results of evaluation. Section 4.3 summarizes the chapter.

4.2 Memory Model Development

This section draws the roadmap to the development of the proposed memory model as shown in Figure 4.1. It includes two memory schemes; the component-based memory scheme and the population-based memory scheme, to address respectively the problems with restarts and local searches within the MMAS. Two algorithmic components have emerged, which are reactive heuristics and recursive local search technique. The components are evaluated as they are the underlines of RMMAS.

Two well-known combinatorial problems are used in evaluation, namely QAP and TSP.

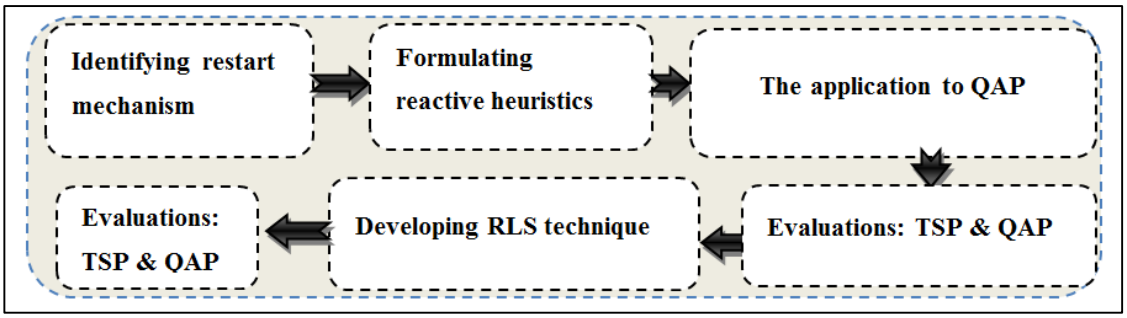


Figure 4.1. The Process for Memory Model Development

4.2.1 Identifying Restart Mechanism

Restart is a generic exploration mechanism. In order to address the problem of arbitrary restart, where the ants re-explore the same regions again and again, it is important to identify how the restart point can be determined effectively. To achieve this goal, let us consider an ACO algorithm suffers stagnation problem where the algorithm is not available for the quality of solutions by time. The said algorithm needs to restart the search so as to escape this situation. The critical issue is to indicate the best moment for restart. This can be done using exploration measures as feedback from the optimization process. The combination of two exploration indicators, the *acceptance criteria* with *branching factor*, is identified to perform the restart. For the acceptance criteria, it is calculated as follows.

$$Restart(S_{gb}, S_k, history) = \begin{cases} -rs & if\ f(S_k) < f(S_{gb}) \\ +rs & if\ f(S_k) \geq f(S_{gb})\ and\ \ i - i_{last} > \epsilon \end{cases} \quad (4.1)$$

where $+rs$ indicates that the convergence happened when the solutions S_k since last best restart i_{last} did not improve for last ϵ iterations (e.g. 250 iterations). For the the average branching factor, it counts the number of factors greater than $\tau_{min} + \Upsilon (\tau_{max} - \tau_{min})$ in the current node in the construction graph, and then counts the average of all the counted factors.

4.2.2 Formulating Reactive Heuristics

Here is the second phase of solving the arbitrary restart in MMAS. The idea is to add new memories to help search agents, i.e. the artificial ants to record the history of their visited neighborhood structures. The ants in this way have two searching states, before and after restart point. In the first state, the ants' search experience is memorized by the proposed component-based memory (CbM) scheme. Figure 4.2 illustrates how the scheme works.

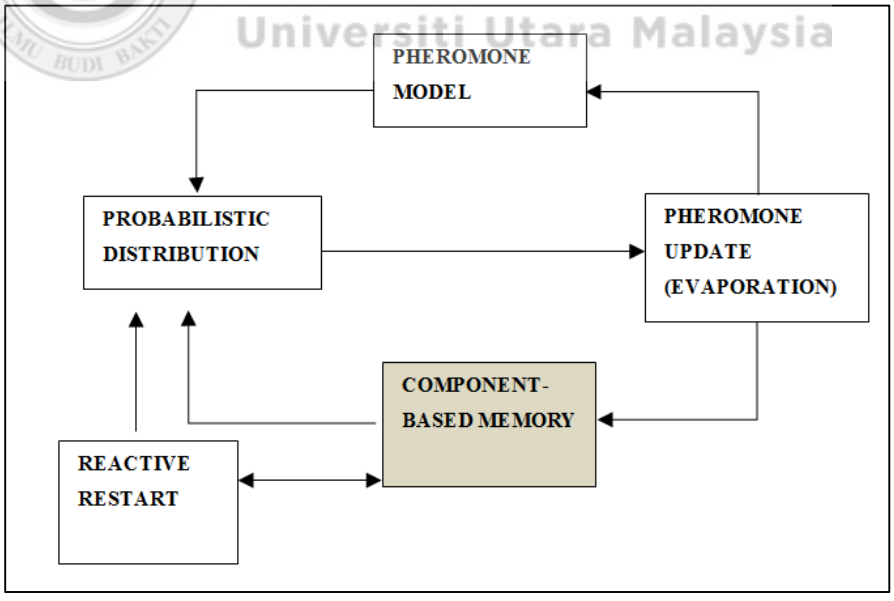


Figure 4.2. The CbM Scheme in Memory Model Development

For some components of solution *s* produced by ant *k*, the pheromone intensity is decreased because of the evaporation influence. The CbM scheme helps in detecting these components to be considered as local heuristics in the probabilistic distribution after restart. Results (see Subsection 4.2.5) showed that this reactive heuristics guides the search for new regions in the search space, and hence improve the exploration behavior of restart strategies. During the optimization process, for some solutions' components, the pheromone intensity is decreased below the predefined threshold (in MMAS denoted by τ_{min}) because of the evaporation model influence. These are the unvisited solutions' components. On-demand heuristics are defined to record the unvisited solution components; these are reactive heuristics (RHs). In particular, the evaporation formula in in Equation 4.2 (Stützle, 1999) is reformulated in the present research to include the ability of memorizing the current search as in Equation 4.3.

$$\tau_{ij} = (1-\rho).\tau_{ij} \quad \forall \tau_{ij} \in T \tag{4.2}$$

$$Evaporate\ (RH, T, \tau_{min}) = \begin{cases} RH \leftarrow rh_1 & if \tau_{ij} < \tau_{min} \\ RH \leftarrow rh_0 & if \tau_{ij} \geq \tau_{min} \end{cases} \tag{4.3}$$

Before the search process is started, RH is initialized to zero (0). The search progress continues together with recording the unvisited components until the next restart point. In this way, unexplored regions in the current search are shifted to the next search.

Once the restart occurs, an improved formula is developed (based on the probabilistic distribution in ACO solution construction function) as described in Equation 4.4, in which the reactive heuristics rh_{ij} is used in the present research.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij}^{\alpha}.\mu_{ij}^{\beta}.rh_{ij}}{\sum_{c_{il}\in N(S^P)} \tau_{il}^{\alpha}.\mu_{il}^{\beta}.rh_{il}} & if\ c_{il} \in N(S^P) \\ 0 \quad otherwise \end{cases}, \tag{4.4}$$

Hence, the ability of ants to remember their previous search influences their future decisions through utilizing three sources of information, which are pheromone trail (τ_{ij}^{α}), pre-heuristics (μ_{ij}^{β}) and reactive heuristics (rh_{ij}), instead of using only τ_{ij}^{α}, and μ_{ij}^{β}. As a result of the aforesaid actions, the proposed RMMAS algorithm is emerged as depicted in Figure 4.3.

Algorithm 4.1: RMMAS
InitializeParameters ()
Initialize_T_Memory ()
Initialize_RH_Memory () *// CbM scheme*
while (**not** terminate) **do**
for *k* := 1 to *m* **do**
if (**no** stagnation) **do**
ConstructSolutions (*T, C*)
else
ReactiveRestart (*S_{gb}, S_k, History*)
ConstructSolutions (*T, C, RH*)
end-else
end-if
$S_{gb} \leftarrow$ argmin{f(S_{gb}), f(S_k | *k*:=1 to *m*)}
Evaporate (*RH, T, τ_{min}*)
DepositPheromone (*T, S^{gb}*)
end-for
end-while
end-algorithm

Figure 4.3. The Pseducode for RMMAS Algorithm

This alternative way of restart is important because of the following reasons. Firstly, the availability of pre-heuristic information is not given in advance for some combinatorial optimization problems such as QAP. Secondly, even if the pre-heuristic information is given, its significant will be decreased subsequently because of the increasing influence of pheromone trail. Thirdly, upon restart and when the ants start to be biased toward high pheromone intensity, the search will stagnate.

Fourthly, comparing the pre-heuristic information which is useless with local search, the results in this chapter verified that the reactive heuristic can play a crucial role in improving the quality of solutions. With the risk of stagnation, together with the ineffectual restart, the reactive heuristic rh_{ij} will be a very useful alternative solution.

4.2.3 The Application to QAP

This section discusses new circumstances in the application of reactive heuristics to combinatorial optimization algorithms with rugged fitness landscape such as QAP. It concerns the situation when local search routines are coupled with RMMAS. For the application, there are two cases: when pre-heuristics information is given in advance such in AS-QAP and ANTS-QAP implementations, and when it is ignored such as in MMAS-QAP implementation (Dorigo & Stützle, 2004). It is well known for the ACO community that the pre-heuristic information is useless with local search. Therefore, it will be omitted in the following implementation.

The QAP can best be described as the problem of assigning a set of facilities (n) to a set of locations (n) with given distances between the locations and given flows between the facilities. The flows and locations are two $n \times n$ matrices dented by A and B respectively, where a_{ij} is the flow between facility i and j, and b_{rs} is the distance between location r and s. The objective is to place the facilities on locations in such a way that the sum of the product between flows and distances is minimal.

When RMMAS is applied to QAP, like other ACO algorithms, the way the solutions are constructed has to be defined first. The way of MMAS in formulating QAP is

followed by RMMAS. It is by assigning facilities in some order to locations. Thus, the pheromone trail τ_{ij} refers to a specific location for facilities, that is, τ_{ij} represents the desirability of assigning facility *i* to location *j*. The ants are used to construct valid solutions for QAP, assigning every facility to exactly one location and not using a location by more than one facility. In this way, a facility is randomly chosen among the still unassigned ones. Then, this facility is put on some free location according to the following probability distribution rule (Dorigo & Stützle, 2004).

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}}{\sum_{l \in U(k)} \tau_{il}}, if\ location\ j\ is\ still\ free \\ 0, otherwise \end{cases} \quad (4.5)$$

where *U(K)* denotes the set of unassigned items. The intuition behind this rule is to prefer the high τ_{ij} values, which are the promising location *j* for facility *i*.

Following the application of MMAS to QAP, RMMAS utilizes the pseudo-random proportional rule (one of the important features in ACS) (Dorigo & Stützle, 2004).

$$j = \begin{cases} \arg max_{l \in U(k)} \{\tau_{il}\} & if\ p \leq p_0\ \ (exploitation) \\ S & otherwise\ \ (exploration) \end{cases} \quad (4.6)$$

where p is a random number uniformly distributed in [0, 1] and S is a random variable with probability distribution given by Equation 4.7. The parameter p_0 controls the exploitation of the accumulated experience reflected in the pheromone trail matrix versus the biased exploration of new solutions.

In the following sections, the performance of RMMAS compared with the original performance of MMAS is presented. The results show that RHs are positively affected by the quality of solutions generated by RMMAS since it outperformed MMAS for short/long execution time on small/large scale instances of QAP.

4.2.4 Experimental Design for Developing Reactive Heuristics

The main goals of the experiments conducted in the development of RHs are: i) identifying effective reaction using various restart strategies; ii) evaluating RHs without local search; and iii) evaluating RHs when RMMAS is coupled with local search. To achieve these goals, TSP and QAP are used in the experiments. Six ACO variants, they are AS and EAS from Dorigo (1992), RAS (Bullnheimer et al., 1997), BWAS (Cordon et al., 2000), ACS (Dorigo & Gambardella, 1997) and MMAS (Stützle & Hoos, 2000), are used in the comparisons for TSP experiments, while only MMAS is used in the comparisons for QAP experiments, as MMAS is better than the other five ACO variants in solving QAP. The 3-opt local search algorithm is used wherever the local search is coupled with ACO in the experiments. CTM, SRM and QSM (see Chapter Three) are used as comparative measures in the evaluation. Wherever the results are indecisive, the non-parametric statistical tests, Wilcoxon and Chi-square, are used to verify the significance of such results. Ten experiments are conducted for each of the TSP and QAP instances. The running time for each experiment is set to 10 seconds. The execution times are proportional to the size and the structure of the instance. The same approach is applied in the work of Gambardella, Taillard, and Dorigo (1999). These time durations are for short-execution. The experiments are conducted on a Windows 8 64-bit operating system,

processor Intel Core i3-3217U with CPU @ 1.80GHz, RAM 4GB. The proposed algorithm is coded in C language. The QAP and TSP instances are selected from QAPLIB (Burkard et al, 1997) and TSPLIB (Reinelt, 1991) repositories respectively.

The parameter settings are selected from Dorigo and Stützle (2004) when ACO variants are coupled with local search for TSP, while the following configuration is followed when local search is excluded. The parameter settings are based on the literature of each ACO variant. The number of ants (m) is equal to the number of cities, except ACS where m is equal to 10. The pheromone intensity (α) and pre-heuristic distance (β) are equal to 1 and 2 respectively for all variants. Evaporation rate (ρ) is 0.5 for AS and EAS; 0.1 for RAS, BWAS and ACS; and 0.02 for MMAS. Some ACO variants have several additional parameters. The settings for these parameters are: RAS: number of ranks (r) are 6; ACS: q_0 is 0.9; local update parameter is 0.1; number of nearest neighbor cities is 20 for all ACO variants. The initial pheromone (τ_0) is set to $1/\rho * C^{nn}$ in MMAS and to $1/n * C^{nn}$ in ACS. In the original papers of AS, EAS, and RAS, it did not exactly define the value of τ_0. Hence, it is set to $1/\rho * \mathrm{C}^{\mathrm{nn}}$.

ACO variants are tested with and without restart (+rs and –rs respectively). Those with restarts used in the experiments are as follows: i) using acceptance criteria with $\epsilon = 250$ and initial pheromone is set to τ_0; ii) using the same setting for acceptance criteria but with initial value equal to τ_{max}; and iii) using acceptance criteria and lambda branching factor with initial value equal to τ_{max}. The parameter settings are selected from the literature of MMAS-QAP where the number of ants (m) is equal to

5; the pheromone intensity (α) is equal to 1; evaporation rate (ρ) is 0.8 and the exploration/exploitation parameter q_0 is equal to 0.5.

4.2.5 Results of Applying Reactive Heuristics

The results of applying RHs to ACO are reported. The results are divided into three parts; these are the results of identifying effective reaction in ACO, the results of evaluating RHs without local search, and the results of evaluating RHs when coupled with local search.

The first part of results reports the impact of RHs of restart reaction. The results showed that the SRM test for AS is worsened with restarts unlike elitist variants that tend to be more exploitative. The best performance with restarts was obtained by MMAS, while BWAS was the worst without restarts as illustrated in Tables 4.1 and 4.2.

Table 4.1

Results of Identifying Effective Reaction using SRM and CTM Tests

ACO variant	SRM				CTM			
	-rs	+rs			-rs	+rs		
		τ_0	τ_{max}			τ_0	τ_{max}	
		Acceptance criteria	Acceptance criteria	Branching factor + Acceptance criteria		Acceptance criteria	Acceptance criteria	Branching factor+ Acceptance criteria
AS	0/10	0/10	0/10	-	-	-	-	-
EAS	0/10	2/10	0/10	-	-	3	-	-
RAS	1/10	2/10	3/10	-	0.1	0.2	0.13	-
BWAS	0/10	0/10	0/10	-	-	-	-	-
ACS	1/10	1/10	1/10	-	0.19	8.6	1.03	-
MMAS	**2/10**	**6/10**	**4/10**	**8/10**	0.67	1.7	2	**0.64**

It is worth mentioning that the quality of solutions is highly influenced if restarts are used. Using dual feedback criteria (i.e. branching factor and acceptance criteria), the

pheromone model of MMAS outperforms the others. The CTM test using dual feedback criteria in MMAS shows some improvement using restarts (0.64 sec). It can be concluded that the way of managing pheromone in the pheromone model is the key for a successful restart. The restart mechanism in these experiments is a straightforward example for reaction. The optimal solution for eil51.tsp is 426 and successful runs equal to the number of tries which are terminated with optimal solution/ number of tries, and the effect of the restarting mechanism in several ACO variants on the way of managing pheromone in the pheromone model is evaluated. The best effect is selected.

Table 4.2

Results of Identifying Effective Reaction using QSM Tests

ACO variant	QSM (Best)				QSM (Mean)			
		+rs				+rs		
	-rs	τ_0 Acceptance criteria	τ_{max} Acceptance criteria	Branching factor + Acceptance criteria	-rs	τ_0 Acceptance criteria	τ_{max} Acceptance criteria	Branching factor+ Acceptance criteria
AS	429	430	431	-	434	436	437	-
EAS	428	**426**	427	-	433	430	431	-
RAS	426	**426**	**426**	-	430	428	428	-
BWAS	450	427	429	-	468	431	435	-
ACS	**426**	**426**	**426**	-	428	430	427	-
MMAS	**426**	**426**	**426**	**426**	427	427	427	**426**

The second part of results is reported in Table 4.3, which displays the results of evaluating RHs in RMMAS without local search. The evaluation is comparison-based where RMMAS is compared with MMAS in TSP. Excluding local search entails testing RH impact under high explorative environment. The quality of solutions measured by QSM in RMMAS is better than RMMAS. The best solutions, the mean of the best solutions within the ten tries are obviously superior to the

MMAS ones. The results show the stability in the performance of RMMAS because of the incorporating RHs.

Table 4.3

Results of Evaluating the Effectiveness of RHs in TSP without Local Search using QSM Tests

TSP	**Optimum**	**MMAS**			**RMMAS**		
instance		**QSM (Mean)**	**QSM (SD)**	**QSM (Best)**	**QSM (Mean)**	**QSM (SD)**	**QSM (Best)**
berlin52	7542.0	7542.0	0.00	7542.0	7542.0	0.00	7542.0
st70	675.0	677.1	1.85	675.0	**676.9**	2.88	675.0
Eil76	538.0	538.6	0.52	538.0	**538.4**	0.52	538.0
pr76	108159.0	108265.0	285.81	108159.0	**108173.9**	**47.12**	108159.0
gr96	55209.0	55671.8	74.00	55601.0	**55560.9**	**71.02**	55434.0
rat99	1211.0	1211.9	0.88	1211.0	**1211.1**	**0.32**	1211.0
KroA100	21282.0	21342.0	52.14	21282.0	**21334.4**	**47.96**	21282.0
KroB100	22141.0	22301.9	30.26	22237.0	**22294.1**	**23.29**	22237.0
KroC100	20749.0	20797.0	69.12	20749.0	**20789.1**	**68.57**	20749.0
KroE100	22068.0	22337.2	148.60	22068.0	**22268.8**	**137.21**	22068.0
rd100	7910.0	7922.5	16.21	7910.0	**7919.9**	**12.49**	7910.0

In fact, results of Table 4.3 reflect the impact of RHs under explorative environment as this experiment is without local search, the exploitation component and the population diversity is high as the TSP fitness landscape is rugged. The effectiveness of RHs is proportional to such ruggedness as more local optima entail more restart recalls. At each restart, the use of RHs as local heuristics provides the transition probabilistic rule of construction solution function (see Equation 4.4) with a sketch of local optima in terms of one's values (see Equation 4.3). In this way, the arbitrary behavior of restart mechanism to escape the stuck in local optima has become steadier resulting in high quality solutions.

The third part of results is reported in Figures 4.4 and 4.5. The y-axis visualizes the quality of solutions measured by QSM (mean), while the x-axis presents each of the MMAS and RMMAS algorithms.

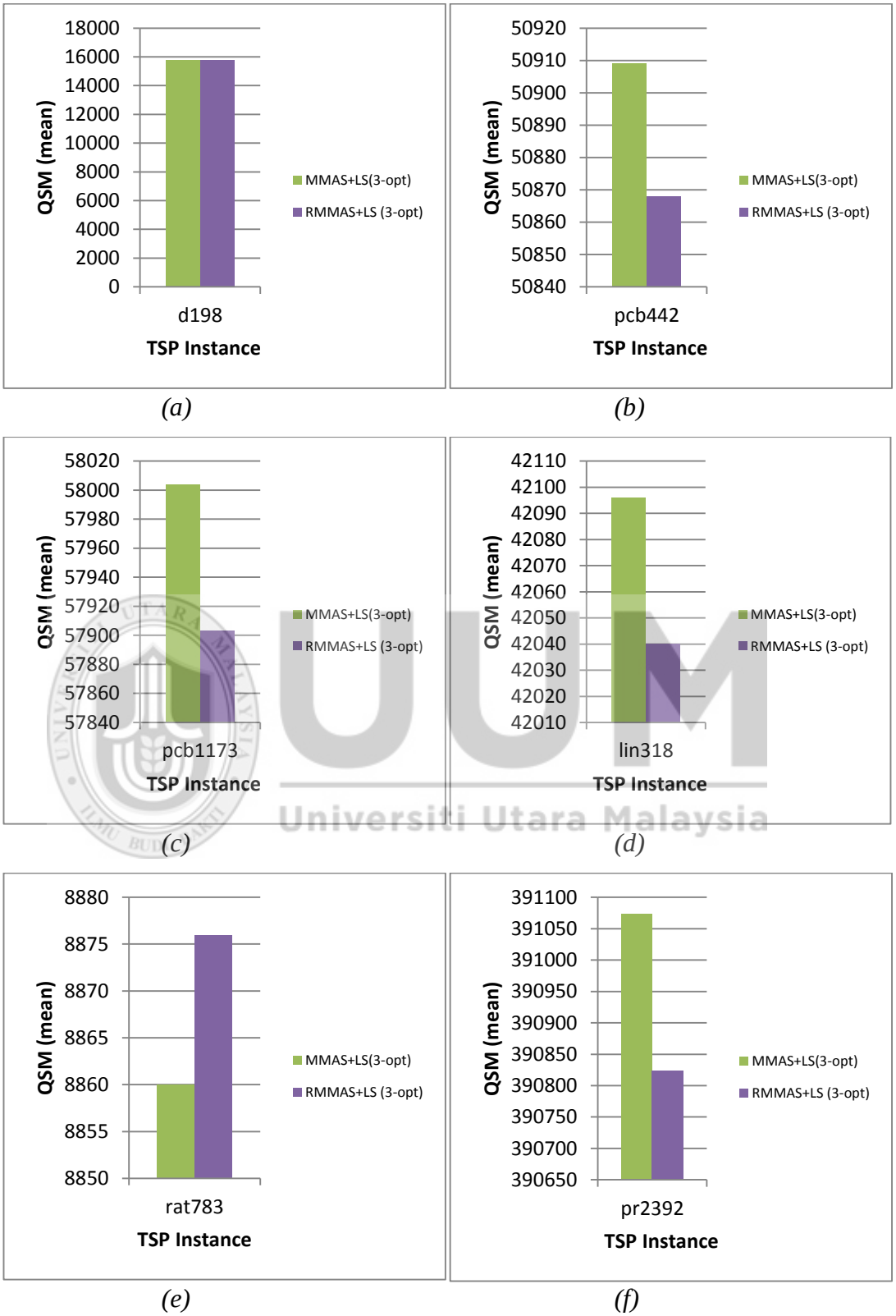


Figure 4.4. Results of Evaluating the Effectiveness of RHs in TSP with Local Search using QSM (Mean) Test

The said algorithms are coupled with 3-Opt local search to solve large TSP instances. This adherence to the 3-Opt way of local search (exploitation component) is to investigate the interrelation between RHs and stochastic local search in large search space. The overall performance of RMMAS outperforms one of MMAS in solving all TSP instances except *rat732.tsp*. However, the RMMAS is not a well-tuned algorithm compared to MMAS.

Again, the effectiveness of RHs has been verified in large search space under the 3-Opt local search circumstance where ants explores distant neighborhoods of the current incumbent solution by conducting three moves from there to a new one if and only if an improvement was made. Using the local heuristics of RHs will help the ants to escape the local optima, if the improvement was not made. It further suggests steady movement to another untraversed neighborhood structure.

In Figure 4.5, the y-axis visualizes the quality of solutions measured by QSM (SD) of the best solutions found during the ten runs conducted to solve TSP instances. The x-axis presents each TSP instance in the performance of MMAS and RMMAS algorithms when coupled with the 3-opt local search. The results showed that the proposed RMMAS has produced good solutions for all TSP instances.

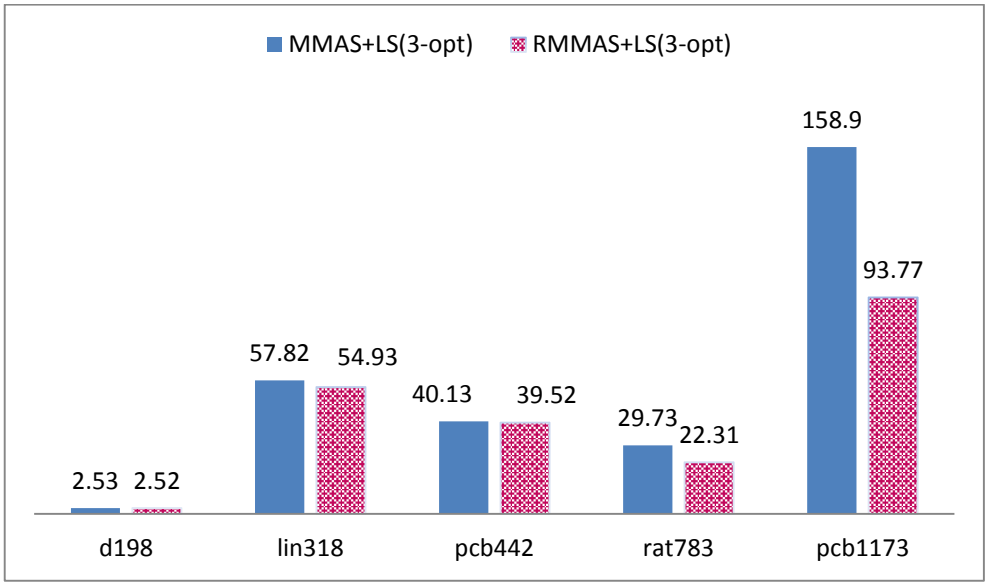


Figure 4.5. Results of Evaluating the Effectiveness of RHs in TSP with Local Search using QSM (SD) Test

For QAP, the results of evaluating RHs in QAP with local search are reported in Table 4.4. The results showed that RHs are useful for solving QAP, especially when pre-heuristics information is not given in advance. However, the results were inconclusive. Therefore, to verify the improvement in a more formal way, the Wilcoxon signed-ranks and Chi-Square statistical tests are performed as in Figure 4.6. Wilcoxon test is performed with 0.05 significance level and one-tailed hypothesis. It is based on the positive and negative ranks of the compared algorithms. The statistical results showed the outperformance of RMMAS over MMAS in the number of ranks. In the comparison of means, MMAS collects (37), while RMMAS collects (135). The p-value is 0.001659. The result is significant at $p \leq 0.05$. In the comparison of standard deviations, MMAS collects (89), while RMMAS collects (101).

Table 4.4

Results of Evaluating the Effectiveness of RHs in QAP with Local Search using QSM Tests

QAP instance	Best known Solution	Seconds	MMAS-QAP			RMMAS-QAP		
			QSM (Mean)	QSM (SD)	QSM (Best)	QSM (Mean)	QSM (SD)	QSM (Best)
bur26a	5426670	8	5427097	636	5426670	**5427083**	**453**	5426670
bur26b	3817852	8	3817935	73	3817852	**3817914**	**47**	3817852
bur26c	5426795	8	5426893	107	5426795	5426906	**113**	5426795
bur26d	3821225	8	3821255	48	3821225	3821305	102	3821232
bur26e	5386879	8	5387074	185	5386879	**5386983**	**136**	5386879
bur26f	3782044	8	3782048	6	3782044	3782048	**4**	3782044
bur26g	10117172	8	10117324	182	10117172	10117642	329	10117208
bur26h	7098658	8	7098708	103	7098658	7098757	163	**7098658**
chr25a	3796	4	4562	172	4304	**4547**	**197**	**4258**
els19	17212548	2	17241610	43635	17212548	17247256	**43395**	17212548
kra30a	88900	8	95609	224	95145	**95600**	313	**94960**
kra30b	91420	9	92298	217	91900	**92248**	277	91910
tai20b	122455319	3	122667105	172642	122455319	**122577869**	184990	122455319
tai30b	637117113	9	638804383	580656	637743822	**638791883**	**447120**	637893225
tai35b	283315445	15	284997173	371182	284180375	**284723472**	372055	**284166043**
tai40b	637250948	24	639646179	677314	638452551	**639486444**	**641417**	638610455
tai50b	458821517	50	461287056	853848	459959918	**461197868**	**637008**	459972346
tai60b	608215054	90	612310940	960895	611081614	**612127094**	**1104145**	**610575173**
tai80b	818415043	225	828968489	3493073	822936304	**828329508**	**2059747**	824542441

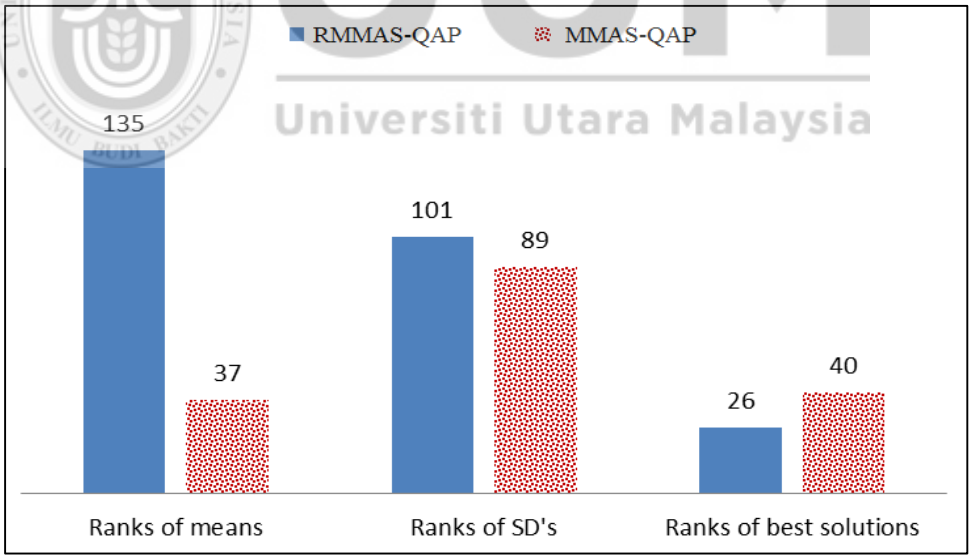


Figure 4.6. Results of Evaluating the Effectiveness of RHS in QAP with Local Search using Wilcoxon Test

The p-value is 0.40517. The result is not significant at $p \leq 0.05$. In the comparison of best solutions, MMAS collects (40), while RMMAS collects (26). The p-value is

0.26763. The result is not significant at $p \leq 0.05$. In the number of ranks for the best solutions, RMMAS did not outperform MMAS, but the overall improvement is not affected. To verify the overall performance, the statistical Chi-square test (see Table 4.5) for frequencies is performed. The significance level used is equal to 0.05. The result is significant at $p < 0.05$ because the p-value is < 0.00001.

Table 4.5

Results of Evaluating the Effectiveness of RHs in QAP with Local Search using Chi-Square Test

Number of ranks	**RMMAS ranks**	**MMAS ranks**	**Row Totals**
Ranks of the means	135 (105.29)	37 (66.71)	172
Ranks of the SD's	101 (116.31)	89 (73.69)	190
Ranks of the best solutions	26 (40.40)	40 (25.60)	66
Column Totals	262	166	428 **(Grand Total)**

So far, the advantage of RHs is to traverse the neighborhood structures drawn by ants. It may be dominated by the neighborhood structures drawn by local search procedures. In this way, the influence of RHs is significant only when local search is not applied. In fact, local search is one of the successful applications for improving the quality of solutions within ACO. Therefore, the recursive local search (RLS) technique has been used to avoid sacrificing neither the RHs nor local search. Its idea is based on reinforcing the influence of local search by solving its incompleteness in transferring the promising solutions found in previous generations of ants to the future generations.

4.2.6 Recursive Local Search Development

In the memory model development, the population-based memory (PbM) scheme is designed to promote continuous aggressive exploitation. This can be done using the

proposed *RLS* technique. It is designed to intensify the search inside the neighborhood structure. After ants generate their solutions, the local search procedures will be used to improve each solution before they enter a fixed size population vector, denoted as *P*. At the same time, the best solutions in the current iteration will be added into the *P*. While the number of added solutions does not exceed the maximum size of *P*, the addition process will continue. Once *P* is full, the old added solution, denoted by *best_old solution*, will be temporarily removed from *P* and entered into local search again. If the quality of the just removed solution is improved by the local search, then it will be added again into the *P* vector; otherwise it will be totally removed. Figure 4.7 depicts the scheme of this functionality.

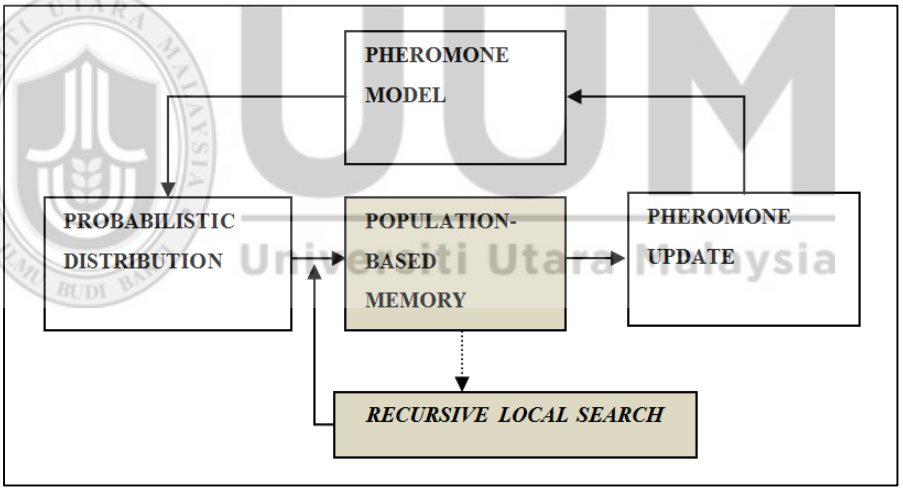


Figure 4.7. The PBM Scheme in Memory Model Development

The RLS technique is designed to overcome the limitation in local search procedures in ACO, where they suffer a premature exploitation because of the incompleteness in transferring neighborhood structures found in the previous search to the next iterations. Figure 4.8 shows the pseducode of the RMMAS algorithm when coupled with the RLS technique. The problem occurs because of the reliance on the current neighborhood structure with ignoring previous structures. In addition to the current

neighborhood structures, this mechanism exploits the *old-best solutions* that are the good solutions in previous structures. The local search procedures are the successful algorithms to traverse the neighborhood structures. Apart from using local search, the ants have their own way to search the neighborhood. Whether the best solution is the best so-far/iteration solution or the old best one, it has to enter a population vector. This recursive way of search neighborhood contributes in a more complete exploitation and is able to produce high quality solutions.

Algorithm 4.2: RMMAS$_{RLS}$
InitializeParameters ()
Initialize_T_Memory ()
Initialize_RH_Memory () *// CbM scheme*
Initialize_P_Memory () *// PbM scheme*
while (**not** terminate) **do**
for k := 1 to m **do**
if (**no** stagnation) **do**
ConstructSolutions (T, C)
else
ReactiveRestart (S_{gb}, S_k, *History*)
ConstructSolutions (T, C, *RH*)
end-else
end-if
$S_{ib} \leftarrow$ argmin{f(S_k | k:=1 to m)}
if (f(S_{ib}) < f(S_{gb}))
$S_{gb} \leftarrow$ argmin{f(S_{gb}), f(S_{ib})}
$S'_{gb} \leftarrow$ LocalSearch(S_{ib})
$S_{gb} \leftarrow$ argmin{f(S_{ib}), f(S'_{ib})}
Add (S_{gb})
if (P = |P|)
$S_{ob} \leftarrow$ Drop ()
$S'_{ob} \leftarrow$ RLS (S_{ob})
$S_{ob} \leftarrow$ argmin{f(S_{ob}), f(S'_{ob})}
else
if(f(S_{ob}) < f(S_{gb}))
Add(S_{ob})
else
Add(S_{ib})
Evaporate (RH, T, τ_{min})
DepositPheromone (T, S^{gb})
end-for
end-while
end-algorithm

Figure 4.8. The Pseudocode for RMMAS Algorithm with RLS Technique

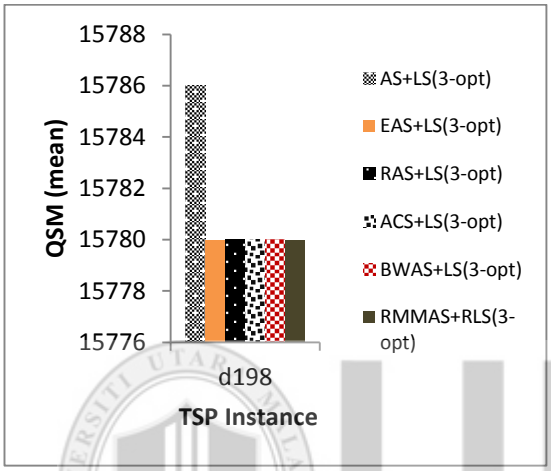
4.2.7 Experimental Design for Developing RLS Technique

To test the influence of the RLS technique on ACO-based local search algorithms, two parts of experimental comparisons need to be conducted. These are to test the influence of RLS on the behavior of the several stochastic local search algorithms when solving the same CO problem, and to test the influence of RLS when solving different CO problems. Five ACO-based local search algorithms are used in the first type of experiments in TSP, while two CO problems are used in the second experiment. The results extend the findings of the previous sections. Ten tries of experiment are conducted for each instance of QAP used in the experiment. The stop condition is proportional to the size and the structure of the instance. The same approach is applied in the work of Gambardella, Taillard, and Dorigo (1999) for long/short execution time. The parameter settings are selected from the literature of MMAS-QAP, where the number of ants (m) is equal to 5; the pheromone intensity (α) is equal to 1; evaporation rate (ρ) is 0.8, and the exploration/exploitation parameter q_0 is 0.5. The metrics that are needed to be tested are the average and standard deviations for finding the best quality of solutions for the ten independent runs, therefore, QSM tests are used as a comparative measure.

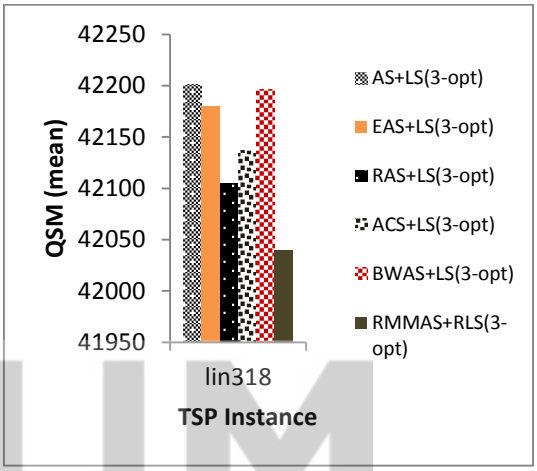
4.2.8 Results of Applying RLS Technique

The influence of the RLS technique on the optimization process is reported into two parts. These are the evaluation against various ACO's performances in one CO problem and the evaluation against various CO problems in one ACO variant. Figures 4.9 (a)-(f) present the first part, while the second part is reported in Figures 4.10 (a)-(f) for TSP, and in Tables 4.6 and 4.7 for QAP.

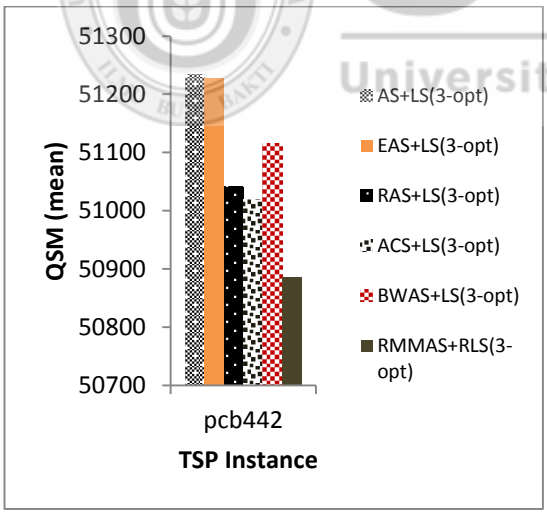
For the first part of results, the y-axis visualizes the quality of solutions measured by QSM of the mean of the best solutions found during the ten runs. The x-axis represents five standard stochastic local search algorithms, namely AS_{LS}, EAS_{LS}, RAS_{LS}, ACS_{LS}, $BWAS_{LS}$. The experiments covered three sizes of TSP instances: small, medium and large.



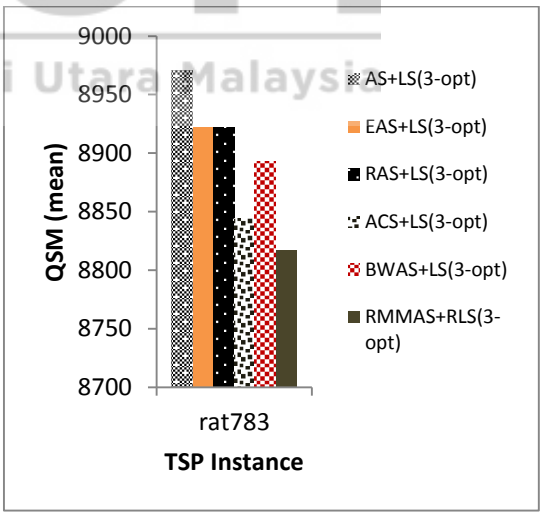
(a)



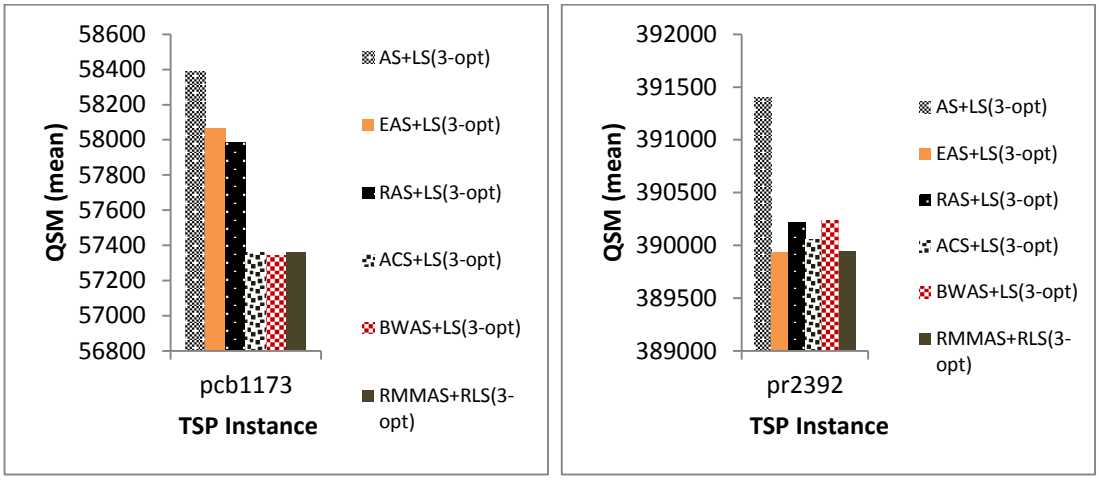
(b)



(c)



(d)



(e) *(f)*

Figure 4.9. Results of Evaluating the Effectiveness of RLS on Various ACO Algorithms using QSM Test

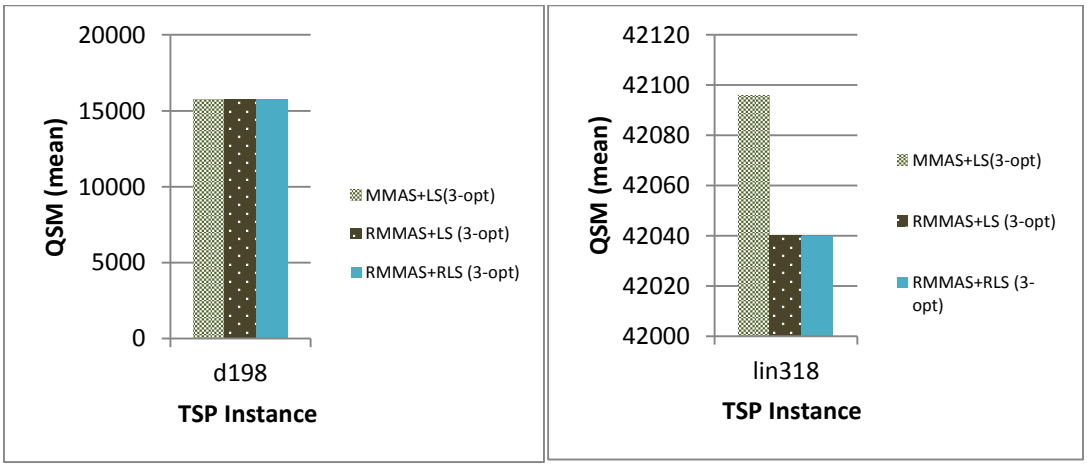
For the small sized instances, the performance of the tested ACO algorithms started equally, except in AS algorithm, which was bad due to the early conference problem (see Figure 4.9 (a)). When the size is increased by using *lin318.tsp*, the performance of the tested algorithms starts to be disparate. In Figure 4.9 (b), the results confirmed the outperformance of the proposed algorithm. The proposed technique is beneficial in solving small scale instances for the TSP problem.

For the medium sized instances, the outperformance of the proposed algorithm continues (see Figure 4.9 (c)-(d)). Therefore, the proposed technique is still beneficial in solving this size of instances for the TSP problem. For the large sized instances, the proposed algorithm showed small outperformance (see Figure 4.9 (e)-(f)). The proposed technique is still beneficial in solving large sized instances for the TSP problem, except in *pcb1173.tsp* instance, in which the results were competitive, but not better than $BWAS_{LS}$. As for the overall performance which is collected and

evaluated by this part of experiments, the proposed RLS mechanism is beneficial in solving all sizes of TSP even with tight run time.

For the second part of results, the previous experiments are extended by considering various CO problems on one hand, and by the comparison with one ACO variant on the other hand. MMAS is considered in the comparison with RMMAS in TSP and QAP. In Figure 4.10, the y-axis visualizes the quality of solutions measured by QSM (mean) of the best solutions found during the ten runs conducted to solve TSP.

In Figures 4.10 (a)-(f), the results showed that the proposed technique outperforms $MMAS_{LS}$ in all TSP instances. In the *pcb442.tsp* instance, the proposed algorithm did not profit from the RLS mechanism. It can be seen from the results that coupling different local search procedures with the MMAS algorithm did not affect the outperformance of the proposed algorithm. That is because of the aggressive exploitative behavior of $RMMAS_{RLS}$ throughout the searching period with the ability to turn to exploration when it is needed.



(a) *(b)*

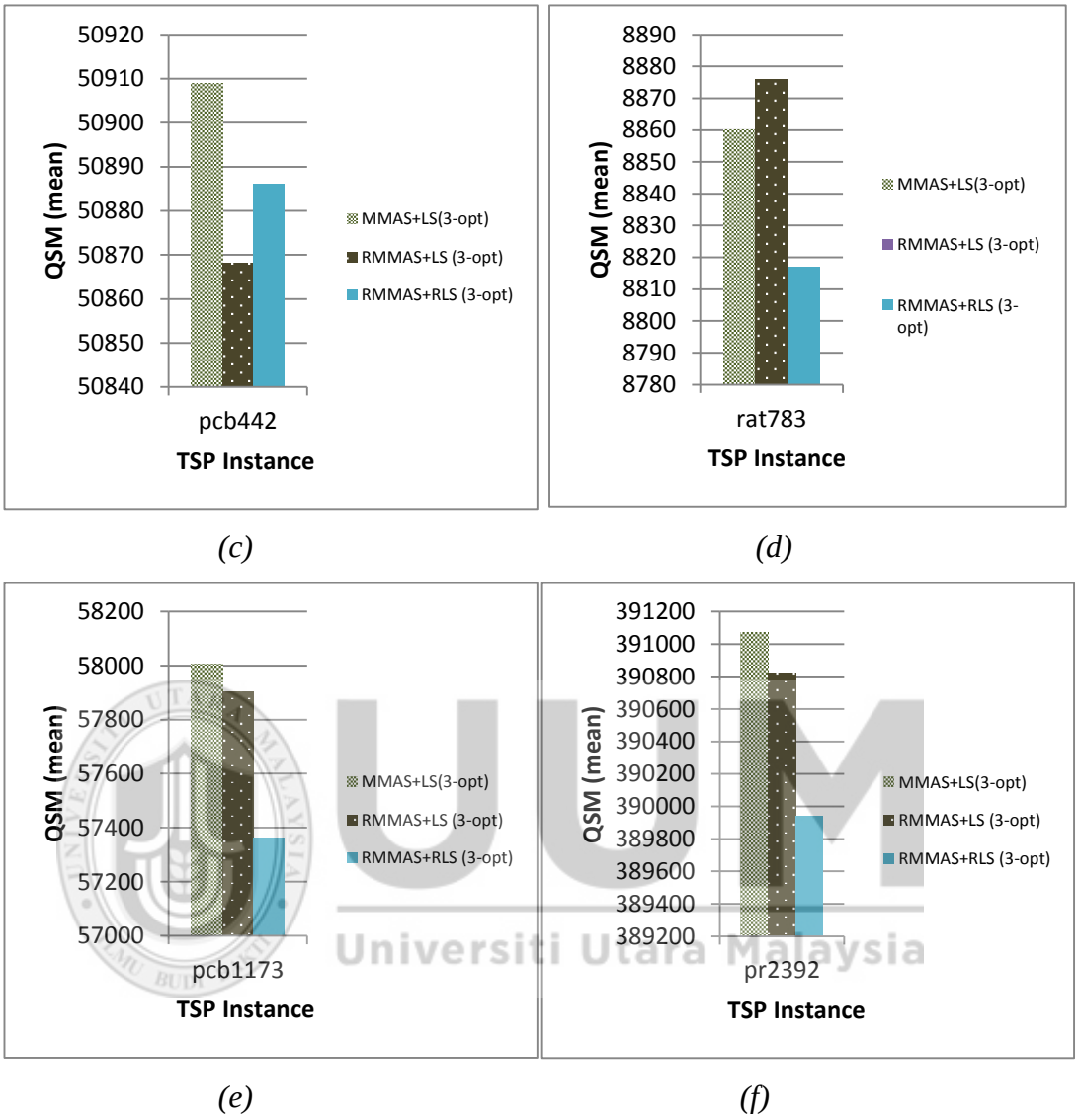


Figure 4.10. Results of Evaluating the Effectiveness of RLS in TSP using QSM Test

For the QAP, the results are reported in Tables 4.6 and 4.7. The results showed that that $RMMAS_{RLS}$ performs better than the original MMAS for the short-runs. The insight that can be concluded is the effectiveness of RLS technique as an exploitation component when coupled with RMMAS for short and long runs.

Table 4.6

Results of Evaluating the Effectiveness of RLS in QAP using QSM Test for Short-Run

QAP instance	Best known Solution	Seconds	$MMAS_{LS}$			$RMMAS_{RLS}$		
			QSM (Mean)	QSM (SD)	QSM (Best)	QSM (Mean)	QSM (SD)	QSM (Best)
bur26a	5426670	8	5427097	636	5426670	**5426670**	**0**	5426670
bur26b	3817852	8	3817935	73	3817852	**3817852**	**0**	3817852
bur26c	5426795	8	5426893	107	5426795	**5426795**	**0**	5426795
bur26d	3821225	8	3821255	48	3821225	3821225	**0**	3821225
bur26e	5386879	8	5387074	185	5386879	**5386879**	**0**	5386879
bur26f	3782044	8	3782048	6	3782044	**3782044**	**0**	3782044
bur26g	10117172	8	10117324	182	10117172	**10117172**	**0**	10117172
bur26h	7098658	8	7098708	103	7098658	**7098658**	**0**	7098658
chr25a	3796	4	4562	172	4304	**4177**	**99**	**3984**
els19	17212548	2	17241610	43635	17212548	**17212548**	**0**	17212548
kra30a	88900	8	95609	224	95145	**94372**	**157**	**94130**
kra30b	91420	9	92298	217	91900	**91523**	**120**	**91420**
tai20b	122455319	3	122667105	172642	122455319	**122455319**	**0**	122455319
tai25b	344355646	5	345428471	762772	344653810	**344379559**	**75620**	**344355646**
tai30b	637117113	9	638804383	580656	637743822	**637218046**	**258852**	**637117113**
tai35b	283315445	15	284997173	371182	284180375	**283768905**	**241686**	**283315445**
tai40b	637250948	24	639646179	677314	638452551	**637375646**	**133920**	**637250948**
tai50b	458821517	50	461287056	853848	459959918	**459293938**	**126779**	**459121468**
tai60b	608215054	90	612310940	960895	611081614	**608922672**	**297495**	**608387539**
tai80b	818415043	225	828968489	3493073	822936304	**822384964**	**1731411**	**820317326**

Table 4.7

Results of Evaluating the Effectiveness of RLS in QAP using QSM Test for Long-Run

QAP instance	Best known Solution	Seconds	$MMAS_{RL}$			$RMMAS_{RLS}$		
			QSM (Mean)	QSM (SD)	QSM (Best)	QSM (Mean)	QSM (SD)	QSM (Best)
bur26a	5426670	50	5426670	0	5426670	**5426670**	0	5426670
bur26b	3817852	50	3817853	4	3817852	**3817852**	**0**	**3817852**
bur26c	5426795	50	5426796	2	5426795	**5426795**	**0**	**5426795**
bur26d	3821225	50	3821225	0	3821225	**3821225**	0	3821225
bur26e	5386879	50	5386879	0	5386879	**5386879**	0	5386879
bur26f	3782044	50	3782044	0	3782044	**3782044**	0	3782044
bur26g	10117172	50	10117172	0	10117172	**10117172**	0	10117172
bur26h	7098658	50	7098658	0	7098658	**7098658**	0	7098658
chr25a	3796	40	4154	116	3946	**4042**	144	**3796**
els19	17212548	20	17212548	0	17212548	**17212548**	0	17212548
kra30a	88900	76	94588	151	94340	**94239**	**148**	**93930**
kra30b	91420	86	91517	88	91420	**91434**	**29**	**91420**
tai20b	122455319	27	122455319	0	122455319	**122455319**	0	122455319
tai25b	344355646	50	344496014	122804	344355646	**344355646**	**0**	**344355646**
tai30b	637117113	90	637612929	440084	637152585	**637128942**	**11091**	**637117113**
tai35b	283315445	147	284231012	101855	284027477	**283378972**	134301	**283315445**
tai40b	637250948	240	638153448	381309	637598806	**637259823**	**19516**	**637250948**
tai50b	458821517	480	460204146	349142	459529895	**459036877**	**59112**	**458923553**
tai60b	608215054	855	610393364	462570	609780832	**608563150**	**104995**	**608387539**

The importance of the proposed technique is achieving the desired balance between RHs as an exploration contributor to the last phase of search and RLS as an exploitation contributor to the initial phase of search. Without RLS technique, the RMMAS algorithm tends to be more explorative and produces suboptimal solutions in short run. Without RHs, the algorithm RMMAS got stuck in some runs as it tends to be more exploitative. The balance is most evident on bur26x instances in which RMMAS succeeded in solve all instances to the optimality in short-run (see Table 4.6). The same concern goes to the application of RMMAS to large instances where exploration and exploitation are needed decisively. The high quality solutions that produced by RMMAS for tai30b, tai35b, tai40b, tai50b and tai60b instances are another evident on the well E&E balance. Without this balance, the problem of premature exploitation induced by the incompleteness of traversing the neighborhood structures impedes the production of high quality solutions.

4.3 Summary

The memory is an essential component in reactive search. This chapter discussed the combination between additional memory features and the distributed computation of ACO. The component-based and population-based memory schemes are two sides of the same coin, which is the memory model. This help in addressing the problems of arbitrary restarts and premature exploitation in local search by the proposal of RMMAS, the new ACO variant. The E&E components of RMMAS, i.e. the reactive heuristics and recursive local search with the help of the memory model, have been evaluated independently using the experimental comparison approach. Different experimental designs have been used to ensure a fair comparison. Empirical and

statistical results have verified the significant improvements in the quality of solutions produced by RMMAS as the exploration and exploitation balance is the profound implication of this high performance. As the exploration versus exploitation is a dynamic strategy, the upcoming chapter elaborates on when and how this dynamism undertakes.

CHAPTER FIVE
EXPLORATION MEASUREMENT AND ADAPTIVE PARAMETERS' SELECTION

5.1 Introduction

This chapter discusses the exploration measurement and the parameters' selection in ACO. Sections 5.2-5.4 propose the so-called *ACO*ustic exploration indicator for the exploration measurement; its experimental design and results. It is important for reactive-based ACO search, tuning an ACO algorithm, online parameter values selection; describing the amount of exploration an algorithm performs, and detecting stagnation situations. Section 5.5 describes the parameters' selection problem. Section 5.6 proposes the strategy of the selection of parameter values during the run. Section 5.7 proposes the strategy of rewarding the promising values. The proposals verified by experiments and results are described in Sections 5.8 and 5.9 respectively. The chapter is summarized in Section 5.10.

5.2 *ACO*ustic for Exploration Measurement

The process of enhancing the exploration measurement is introduced by proposing the *ACO*ustic indicator. The idea of indication is inspired from the acoustical mimicry in the ants-parasites systems. The schema of this iterative process in nature is modeled as shown in Figure 5.1.

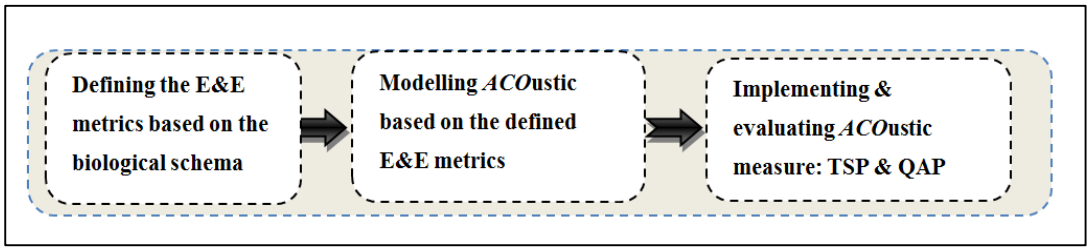


Figure 5.1. The Process of the Modeling and the Implementation of *ACO*ustic

The *ACO*ustic mechanisms are implemented and evaluated within the ACO algorithm. *ACO*ustic is a statistical machine learning tool inspired by the acoustic reaction in nature. So far, utilizing traditional measures, such as acceptance criteria, average λ-branching factor, entropy-based measures (Colas & Monmarch, 2008), or similarity ratio (Solnon & Fenet, 2005) do not satisfy the requirements of reactive search. They are simply not machine learning methods, except the exploration measure developed by Pellegrini et al., (2009). It utilizes agglomerative clustering to quantify the exploration as the number of clusters of solution visited as follows.

$$E\ (B, I, R, h) = |\ L(B, I, R, h)| \tag{5.1}$$

where, L is the set of clusters resulting from the solutions visited by the algorithm B when solving the instance I using the resources R and the seed h. Given such definition, two closest clusters can be concluded when the distance between them is greater than a predefined threshold ϵ_x where $x\%$ of their arcs does not exist in the cluster. However, the definition needs to be reconsidered in terms of robustness against various circumstances.

According to Pellegrini and Favaretto (2012), such threshold must be coherent with the magnitude of the distance matrix. In TSP, ϵ_x = 7.8 in MMAS without local search, ϵ_x = 17.5 in MMAS with 2-opt local search and ϵ_x = 35.8 with 3-opt local search. The situation changes with the change of circumstances. For example, ϵ_x = 1.003 when it is applied to genetic algorithms with no local search, ϵ_x = 16.56 with 2-opt local search, and ϵ_x = 46.5 with 3-opt local search. Therefore, this situation leads to an unstable measurement, especially when more rugged CO problem's instances, such as QAP, need to be solved by algorithm *B*. This problem has been solved in *ACO*ustic. To present the overall idea, the following subsections discuss: the biological schema; modeling *ACO*ustic, the implementation and evaluation.

5.2.1 The Biological Schema

Rapid and effective communication between ants is a key attribute that enables them to live in dominant, fiercely protected societies. *Myrmica* ant colonies, in particular, are exploited by social parasites called *Maculinea* butterflies (Barbero et al., 2012). The process of *Trophallaxis* (i.e. distributing liquid food from the 'social stomach') between attendance worker and other nest-mates is the main process in the food foraging behavior of ants. The worker ants produce acoustics during the process. The *Maculinea* larvae interfere with the *Myrmica* system and produce similar acoustics to that of the colony. The high number of worker ants leads to a low relatedness between nest-mates. A greater variance in nest-mates' acoustic signals leads to a higher likelihood of being infested (Barbero, Thomas, Bonelli, Balletto, & Schönrogge, 2009). Through this indicator, the larva can decide the optimal point to leave the colony before it is discovered by other ants.

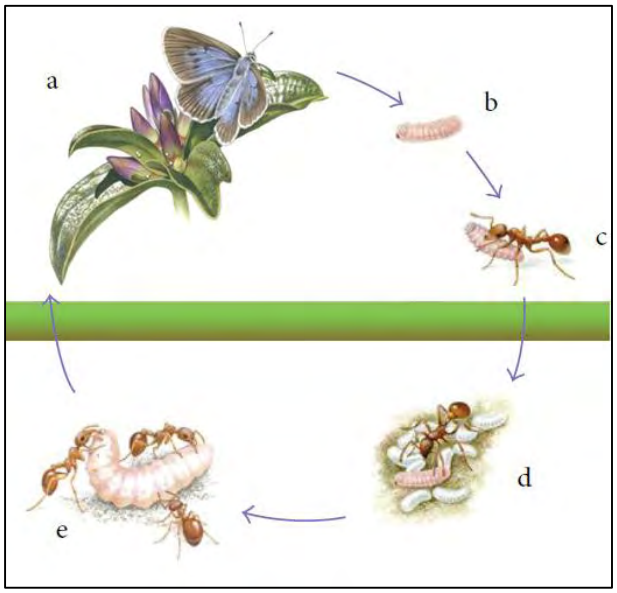


Figure 5.2. The Ants- Parasites System

This given social membership in ants-parasites system, i.e. the *Myrmica-Maculinea* system, includes sharing resources such as the process of regurgitating and distributing liquid food in their 'social stomach' to other hungry nest-mates as illustrated in Figures 5.3 (a) and (b).

(a) *(b)*

Figure 5.3. (a) The Trophallaxis and Antennation between Ants (b) Trophallaxis between Ants and Parasites

Myrmica workers frequently stridulate during the trophallaxis process. The stridulatory signal is simple and contains one type of massage, such as "food is exhausted". *Myrmica* queens can generate distinctive sounds to reinforce their supreme social status. The *Maculinea* larva interferes with this system and produces similar sounds to that of the queen (Barbero et al., 2012) as shown in Figure 5.4.

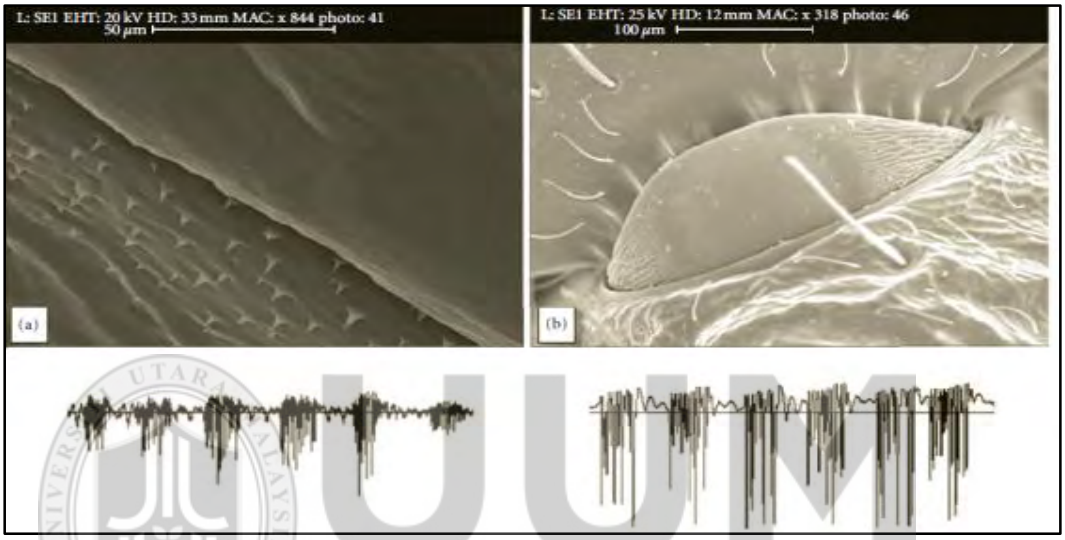


Figure 5.4. The Morphology (Upper Part) and Sounds (Lower Part) of the Acoustical Organs of (a) Parasites Queen and (b) Ant Queen

The larva is able to evaluate the situation inside the nest whether to leave or stay. If the relatedness between nest-mates becomes high, then the likelihood of being clustered around the larva will become low. This is an indication to the larva to explore another nest before being killed; otherwise the larva will continue to exploit the current nest until further notice. The acoustic reaction in this process can be simplified in three basic components as shown in Figure 5.5.

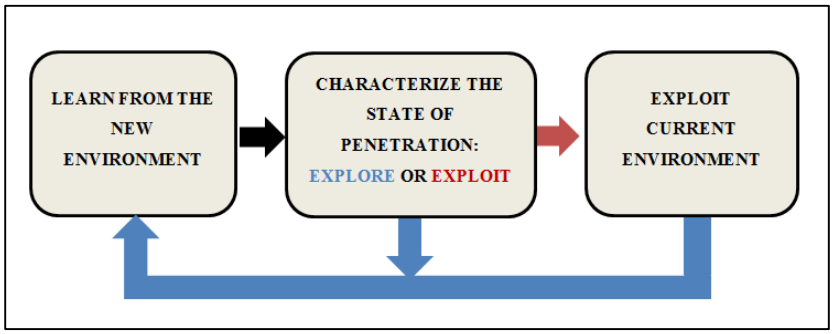


Figure 5.5. The Scheme of Acoustical Indication in Nature

In ACO modeling, the characteristics of artificial ants' are inspired from the real ants' foraging behavior. The construction graph simulates the environment that ants and larvae agents are moving on. For larvae agents, the interaction with the new environment is highly related with the state of penetration, i.e. the learning process. The agents can decide whether to continue with the current exploitation or to explore another environment. To simulate the process of characterizing the state of penetration, statistical analyzing and the agglomerative clustering algorithms are developed in this chapter.

5.2.2 Modeling *ACO*ustic

In this subsection, the way of characterizing the state of penetration is used as a didactic tool to explain the idea behind the *ACO*ustic's proposal. The behavior of the ACO algorithm describes in terms of the exploration and exploitation processes. According to the scheme described in Figure 5.5, the natural scheme in parasites-ants system translates into problem-solving models as follows.

Let a construction graph $G = (N, A)$ represent a CO problem, where N is the set of nodes; A is the set of arcs; $|A| = a$ and $|N| = n$. The fitness landscape of the given CO

problem is defined by: P is a population set which includes all solutions to the CO problem, where each solution $s \in P$ is assigned a fitness value *f(s)*; and has a structure of neighborhood $N \subseteq P \times P$. A colony of artificial ants performs a biased walk in this landscape with the goal of finding low *f(s)* (in the case of minimization problems). The set $C_p(t)$ represents the collection of acoustics (sounds) that emanates from the landscape traversed by the ants of a perspective colony at time *t* where $C_p(t) \subseteq P(t) \times P(t)$ where c_i and c_{i+1} are two acoustics belonging to $C_p(t)$ where $c_i = \{x_1, x_2, \ldots, x_a\}$; $c_{i+1} = \{y_1, y_2, \ldots, y_a\}$ where the long signal of each acoustic is equal to *a*. The relatedness between two nest-mates is defined by the similarity between their acoustics. Two acoustics c_{i+1} and c_i are considered as similar if their similarity neighborhood *SN* is below a predefined threshold *X*.

$$SN\left(c_{i+1}, C_p(t)\right) = \min_{s \in C_p(t), t = 0..current\ iteration\ and\ c_{i+1} \neq c_i} d(c_{i+1}, c_i) \quad (5.2)$$

$$SN\left(c_{i+1}, C_p(t)\right) > X \quad \text{exploration} \quad (5.3)$$

$$SN\left(c_{i+1}, C_p(t)\right) \leq X \quad \text{exploitation} \quad (5.4)$$

where *d* is the Euclidian distance between two acoustics in $C_p(t)$ within Euclidian space R^n. Exploration occurs when *SN* of the two acoustics is greater than the boundary of the neighborhood threshold (*X*), otherwise, it is identified as exploitation.

A population-based memory scheme is used to record the best-iteration solutions produced by the algorithm during the run. An agglomerative clustering procedure is applied to the recorded population every ten iterations. This is to determine the similarity features of the population through its acoustics during the past ten

iterations. A matrix of distances is defined to conduct the clustering, and then to detect the number of clusters. The Euclidean distance d between c_i and c_{i+1} is a common way for finding similarity as follows.

$$d(c_i, c_{i+1}) = \sqrt{\sum_{j=1}^{a} (x_j - y_j)^2} \quad (5.5)$$

The quantity d may have different magnitudes so that it is normalized to the size of the population.

$$d_{norm} = d/|P| \quad (5.6)$$

Three statistic medians (mean, variance and standard deviation) are derived in (5.7), (5.8) and (5.9) respectively.

$$mr(t) = \sum_{i=1}^{max-1} \sum_{j=i+1}^{max} D_{ij} \Big/ ((max^2 - \max)/2) \quad (5.7)$$

$$vr(t) = \sum_{i=1}^{max-1} \sum_{j=i+1}^{max} (C_{ij} - mr)^2 \quad (5.8)$$

$$stdr(t) = \sqrt{vr/(max - 1)} \quad (5.9)$$

where D_{ij} is the normalized distance between two acoustics and max is the maximum size of the distance matrix. In order to minimize the computational efforts and keep the algorithm non-weights the size of matrix fixes to ten, the agglomerative hierarchical technique is used for calculating the number of clusters as follows.

$$C_{Num}\left(C_p(t)\right) = \left| L\left(C_p(t)\right) \right| \quad (5.10)$$

where L is the set of clusters resulting from the solutions that are visited by the ants. The statistics and clustering information are combined. The relatedness between ants can be calculated by finding the difference between the mean of distances and the number of clusters by the standard deviation of distances as follows.

$$rltdnss = (mr - C_{Num})/stdr \tag{5.11}$$

The definition of exploration and exploitation in (5.3) and (5.4) can be reformulated based on the relatedness between acoustics ($rltdnss$) as follows.

$$rltdnss > X_{rltdnss} \quad (exploration) \tag{5.12}$$

$$rltdnss \leq X_{rltdnss} \quad (exploitation) \tag{5.13}$$

where $X_{rltdnss}$ denotes the lowest degree of relatedness. It is detected by capturing the first value of $rltdnss$ within the first ten iterations. For instance, when C_{Num} is decreased from 10 to 8, this indicates that what is occurring at this moment is exploitation. In contrast, if C_{Num} stays as it is, this indicates the exploration is high. The assignment of $X_{rltdnss}$ has to be complete within the first ten iterations. Hereafter, each new value of $rltdnss$ will be characterized as either exploration or exploitation accordingly.

5.2.3 *ACO*ustic Implementation

This subsection walks through the implementation of the *ACO*ustic algorithm. The pseudocode of the algorithm is illustrated in Figure 5.6. The nearest neighborhood threshold X is entered, the vector of acoustics clusters C_i is defined and other

variables such as *miniDist*,C_{Num}, $X_{rltdnss}$ and *max* are initialized as in Figure 5.7. Following the biological way of finding similarities between acoustical signals made by individual queens and workers (Thomas, Schonrogge, Bonelli, Barbero, & Balletto, 2010), *findSimilarities* algorithm generates the matrix of Euclidean distances between artificial acoustics. Next, the statistical medians are calculated as in Figure 5.8.

Algorithm 5.1: *ACO*ustic **()**
*initialization***()**
    while (not *terminate***()) do**
    {mr, stdr}=findSimilarities **()**
    rltdnss = determineRelatedness **(***mr, stdr***)**
end- while
return *rltdnss*
end- algorithm

Figure 5.6. The Pseudocode of Acoustic Algorithm

procedure *initialization***()**
Input: *X*
Define: $C = \{C_1, C_2,.., C_{|a|}\} = \{\{c_1\}, \{c_2\},.., \{c_{|a|}\}\}$
Initialize*: miniDist*, C_{Num}, *max*
end- procedure

Figure 5.7. The Initialization Procedure

Algorithm 5.2: *findSimilarities* **()**
foreach C_h, $C_k \in C$ **do**

$$D = d_{c_h, c_k} = \sqrt{\sum_{i=1}^{a} (x_i^h - x_i^k)^2}$$

end-foreach

$$mr = (\sum_{i=1}^{max-1} \sum_{j=i+1}^{max} (D_{i,j}/m)/((max^2 - max)/2)$$

$$vr = \sum_{i=1}^{max-1} \sum_{j=i+1}^{max} (C_{ij} - mr)^2$$

$$stdr = \sqrt{\frac{vr}{max - 1}}$$

end- algorithm

Figure 5.8. The Pseudocode of Find similarities Algorithm

In *determineRelatedness*, the minimum distance *miniDist* is calculated from the distance matrix that is generated earlier (Figure 5.9). The nearest two clusters are united, the distance matrix is recalculated, and finally *miniDist* and the number of clusters C_{Num} are updated. In Figure 5.9, the number of clusters and the statistics collected earlier are combined and returned as a relatedness quantifier denoted as *rltdnss*.

Algorithm 5.3: *determineRelatedness* **()**
miniDist = $\min_{c_h, c_k \in C} d_{c_h, c_k}$
repeat
  $C_h = C_k \cup C_h$
  $C = C \setminus \{C_k\}$
  foreach $C_w \in C \setminus \{C_h\}$ **do**
    $d_{c_h, c_w} = d_{c_w, c_h} = \min\{d_{c_h, c_w}, d_{c_k, c_w}\}$
  end- foreach
  miniDist = $\min_{c_h, c_k \in C} d_{c_h, c_k}$
  nc = | *C* |
until (*miniDist* $\leq X$ **)**
rltdnss ← *mr* – C_{Num}/ *stdr*
return *rltdnss*
end- algorithm

Figure 5.9. The Pseudocode of Determine Relatedness Algorithm

5.3 Experimental Design for Developing *ACO*ustic

In this Section, *ACOustic* is applied for several standard ACO algorithms under various conditions. The implemented algorithms are AS, EAS, ACS, RAS, MMAS, BWAS. The aim of the application is: i) to examine the ability of *ACO*ustic for monitoring the exploration behavior of ACO algorithms when searching two different fitness landscapes: TSP and QAP; and ii) to evaluate its performance against the state-of-the-art measurement tool in ACO. Its performance is reported to be compared with average λ-branching measure for TSP and with exploration measure for TSP and QAP. In the former comparison, the effect of the parameters of

MMAS algorithm on the exploration and exploitation mechanisms is analyzed. Several scenarios have been considered. In the latter comparison, the effect of the raggedness of fitness landscape is analyzed. The shape of the landscape of TSP versus the one of QAP is considered.

The experimental setting follows the literature. The parameters analyzed are α, β, ρ *and m*. The parameter setting suggested by Pellegrini et al., (2012) has been considered. The stopping criteria considered is either the completion of 350sec (only the first 3000 iterations are reported) for large instances or finding the optimal solution for small instances. Using restarts and local search are denoted by *+rs* and *+ls* respectively. Using the same symbols with the minus sign gives the opposite meaning. The C coding is used in the implemented algorithms. The experiments are conducted on a Windows 8 64-bit operating system, processor Intel Core i3-3217U with CPU @ 1.80GHz, RAM 4GB. Each experiment is executed ten times to avoid the stochastic behavior. The main results of this application are figured as below. The TSP instances used in the experiments are selected from TSPLIB repository and from the 8th DIMACS challenge. Following the TSPLIB format, d198 instance is selected. Following the DIMACS format, one random instance is generated using *portgen*, the instance generator adopted in the 8th DIMACS TSP challenge. It is generated with *size = 2000* and *seed = 39200.* The *kra30a.qap* instance used in the experiments is selected from the QAPLIB repository.

5.4 Results of *ACO*ustic's Application

The general performance of *ACO*ustic is analyzed. The computational results are twofold. The first part of results reported the robustness of the proposed tool against

the difference in the raggedness of fitness landscapes (Figures 5.10-5.20). The second part of results reported the ability of the proposed tool to analyze the convergence behavior of ACO algorithms against different CO problems (Figure 5.21) and different parameter settings (Figure 5.22).

In Figures 5.10-5.15, the y-axis visualizes the exploration while the y-axis represents the number of objective function evaluations in TSP. The ability of *ACO*ustic to provide the same exploration insights of the λ-branching measure and the exploration measure is tested. Results showed that *ACO*ustic is able to draw the same shape of these measures. Since the neighborhood threshold is mutual characteristic between *ACO*ustic and exploration measure, it will act as a mirror to reflect the robustness of each of them against the change in the value of the threshold and the change the ruggedness in the fitness landscape. For the neighborhood threshold of *ACO*ustic, the statistical information gathered (*mr* and *stdr* of relatedness, see Equation 5.11) are combined with the number of clusters produced by the agglomerative clustering procedure included in *ACO*ustic algorithm. With high ruggedness landscape, both parts of the Equation 5.11 give the same contribution. Therefore, one cannot find a slight difference between *ACO*ustic and exploration measure. The comparison with λ-branching measure follows the same concern as it is statistical indicator.

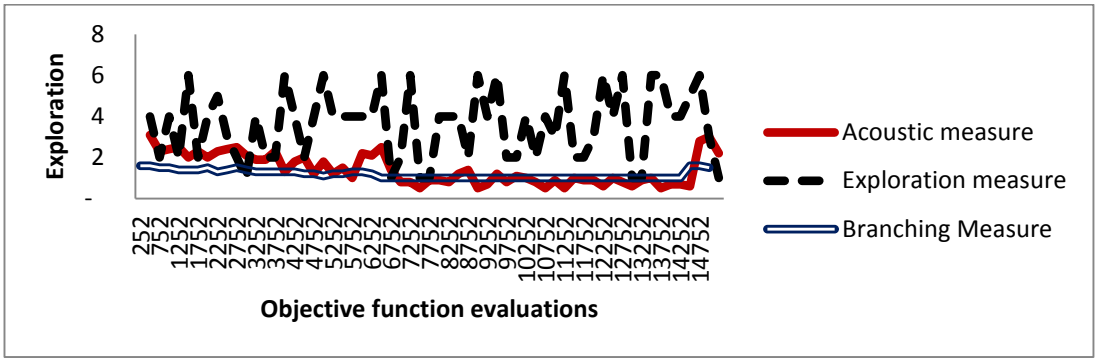


Figure 5.10. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 8

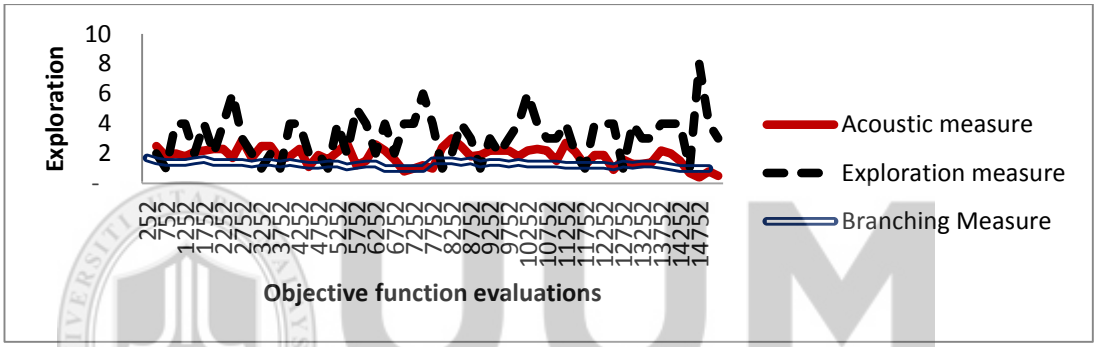


Figure 5.11. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 7

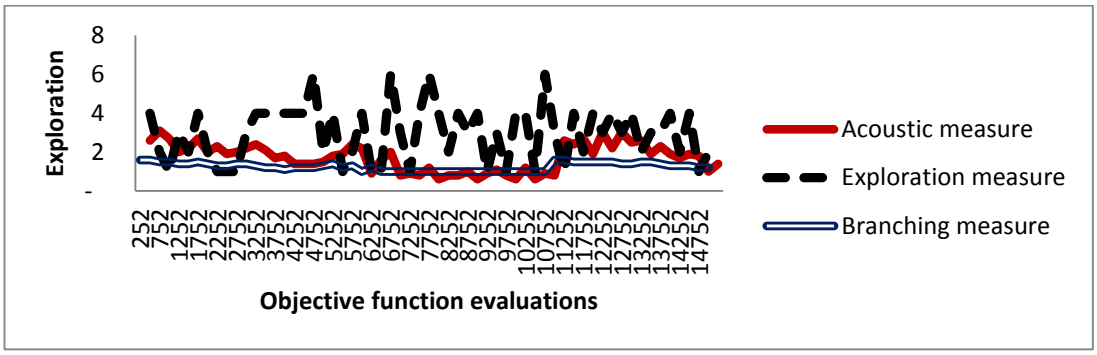


Figure 5.12. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 6

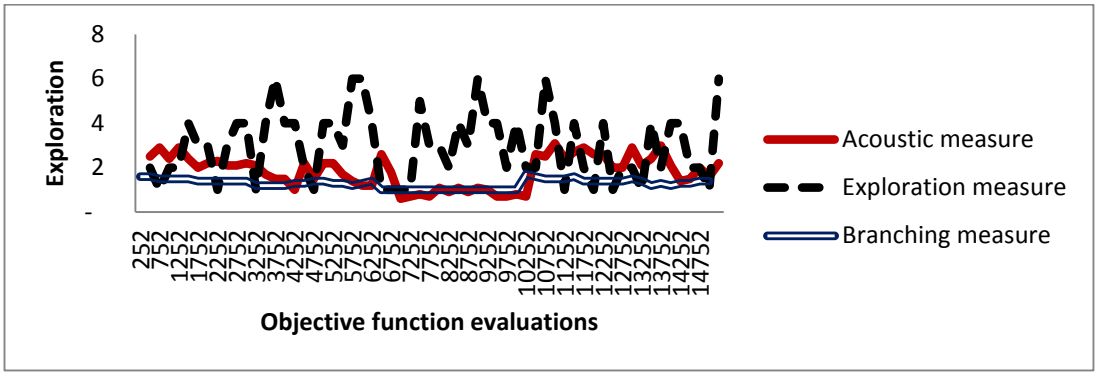


Figure 5.13. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 5

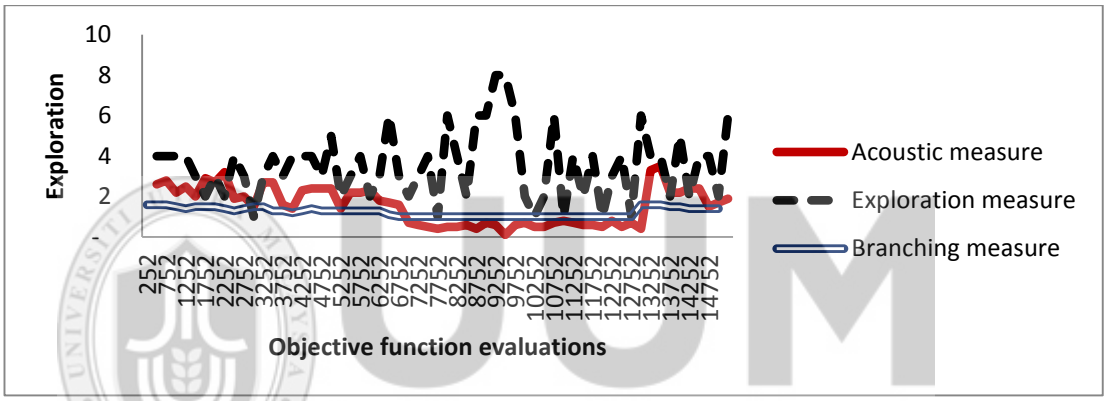


Figure 5.14. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 4

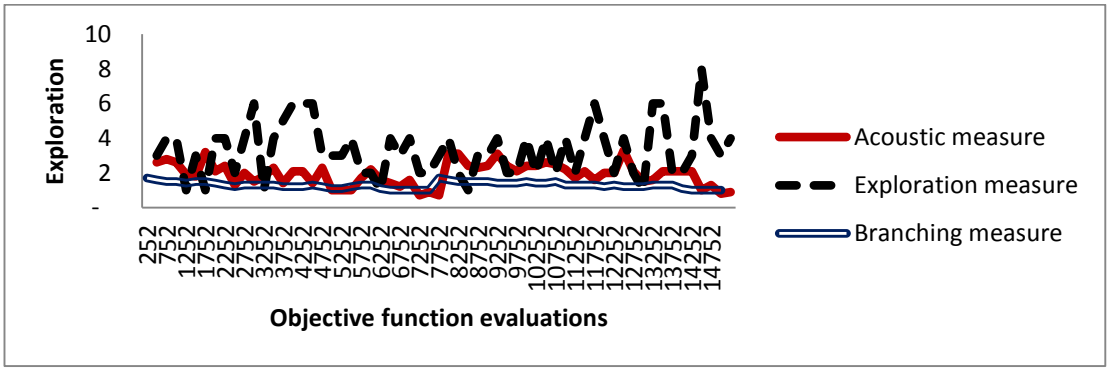


Figure 5.15. Results of Comparing Acoustic with other Exploration Measures in TSP with Nearest Neighborhood Threshold = 3

In Figures 5.16-5.20, the y-axis visualizes the exploration while the y-axis represents the number of objective function evaluations in QAP. The robustness of *ACO*ustic against two circumstances of reading the exploration behavior is compared to the λ-branching measure and the exploration measure is tested. Results showed that using different values of neighborhood threshold *ACO*ustic gives different insights than other measures. For the λ-branching, when the fitness landscape flattens, the statistical analysis becomes fruitless because of the high similarity ratio between the solutions. For exploration measure, the neighbourhood threshold will be of decisive importance in the comparison (see Figure 5.19). In contrast, *ACO*ustic dedicates the first 10 iterations to calculate automatically the relatedness value (see Equations 5.12 and 5.13). In this sense, it can adapt easily for the change in the ruggedness, which means it is able to indicate the diversity of population regardless of similarity ratio between the solutions. The results showed significant robustness against the difference in the value of neighborhood threshold with flatten fitness landscape.

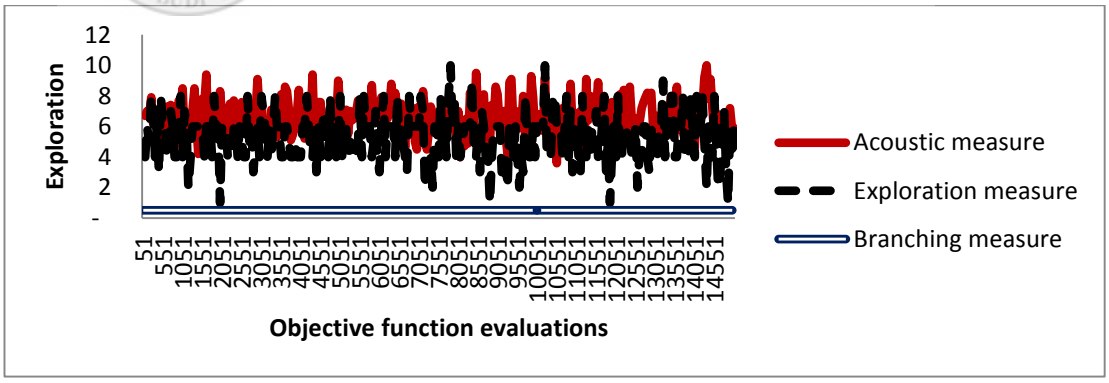


Figure 5.16. Results of Comparing Acoustic with other Exploration Measures in QAP with Nearest Neighborhood Threshold = 8

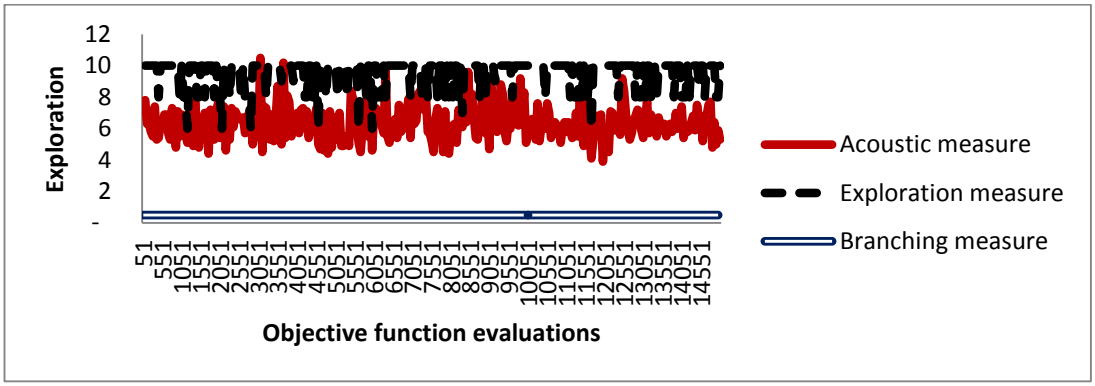


Figure 5.17. Results of Comparing Acoustic with other Exploration Measures in QAP with Nearest Neighborhood Threshold = 7

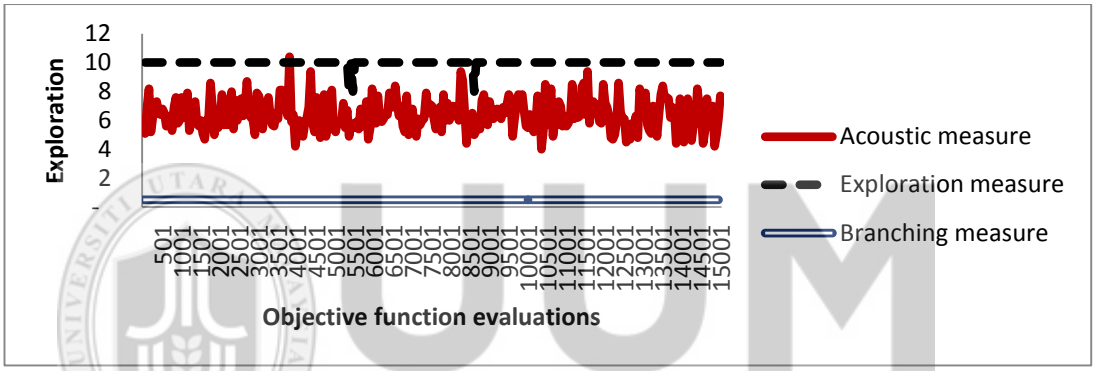


Figure 5.18. Results of Comparing Acoustic with other Exploration Measures in QAP with Nearest Neighborhood Threshold = 6

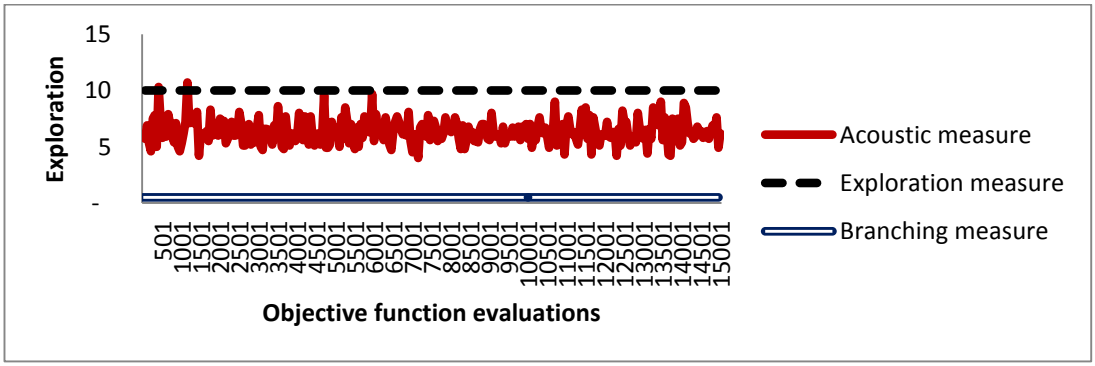


Figure 5.19. Results of Comparing Acoustic with other Exploration Measures in QAP with Nearest Neighborhood Threshold = 5

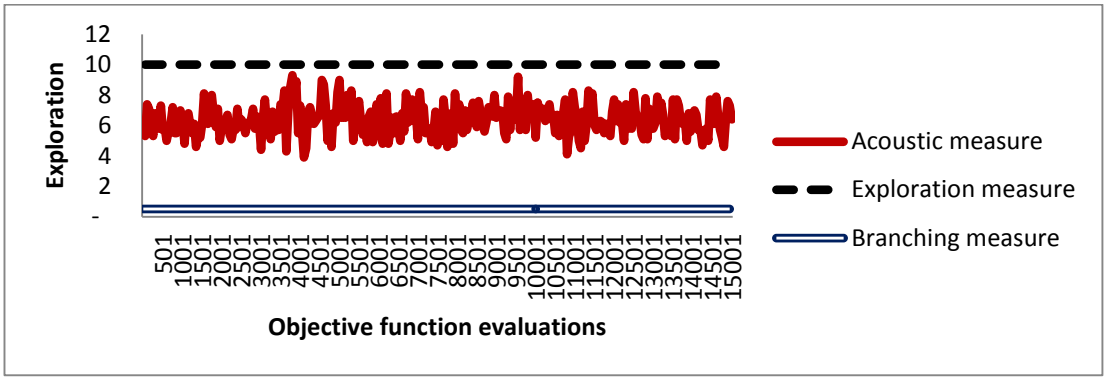
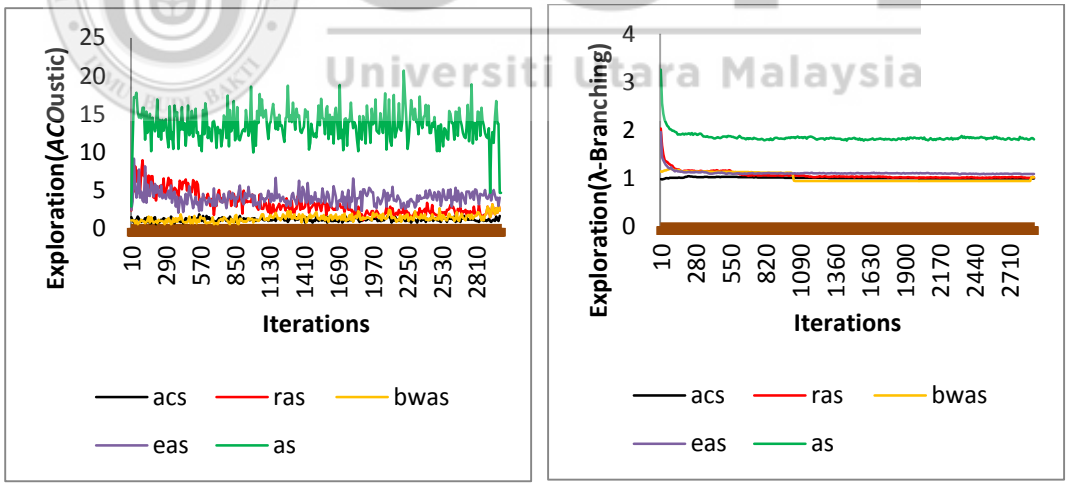


Figure 5.20. Results of Comparing *ACO*ustic with other Exploration Measures in QAP with Nearest Neighborhood Threshold = 4

In Figures 5.21 and 5.22, the y-axis visualizes the exploration using *ACO*ustic measure comparing with λ-branching measure. The y-axis presents the number of iterations. The general performance of five ACO algorithms, namely AS, EAS, RAS, ACS and BWAS, is reported in Figure 5.21. In Figure 5.22 the effect of parameters on MMAS behavior is reported.



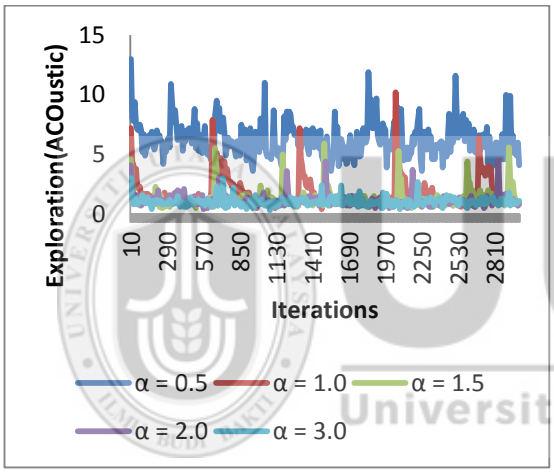
(a) *(b)*

Figure 5.21. Results of Utilizing *ACO*ustic (a) against Branching Factor (b) to Evaluate Various Exploration Behaviours in TSP

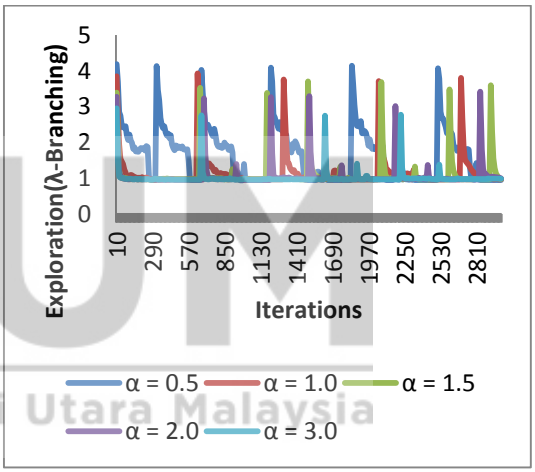
In Figure 5.21, the results of analyzing using the proposed measure showed that AS tends to be a very explorative algorithm. The rest of the tested algorithms either start

with a very short exploration phase followed by a very aggressive exploitation phase (e.g. EAS and RAS) or skip the initial exploration phase (e.g. ACS and BWAS). This is mainly achieved by a stronger emphasis given to the best tours found during the search.

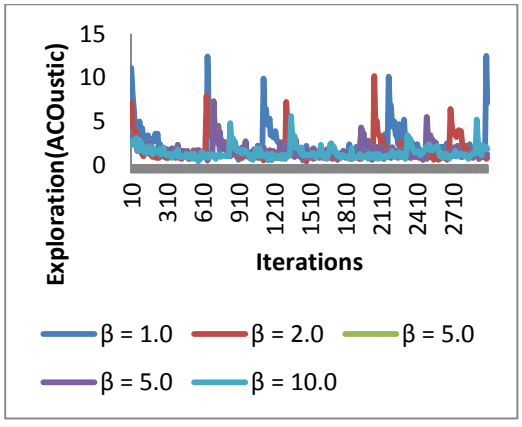
In general, when compared with λ-branching measure, the proposed measure draws the same shape for TSP. The same insights are gathered when the exploration behavior is influenced by parameter tuning.



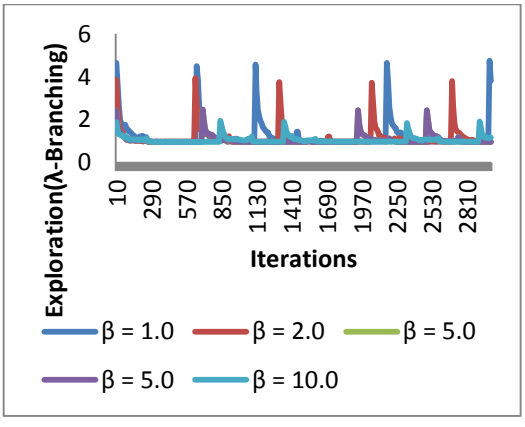
(a)



(b)



(c)



(d)

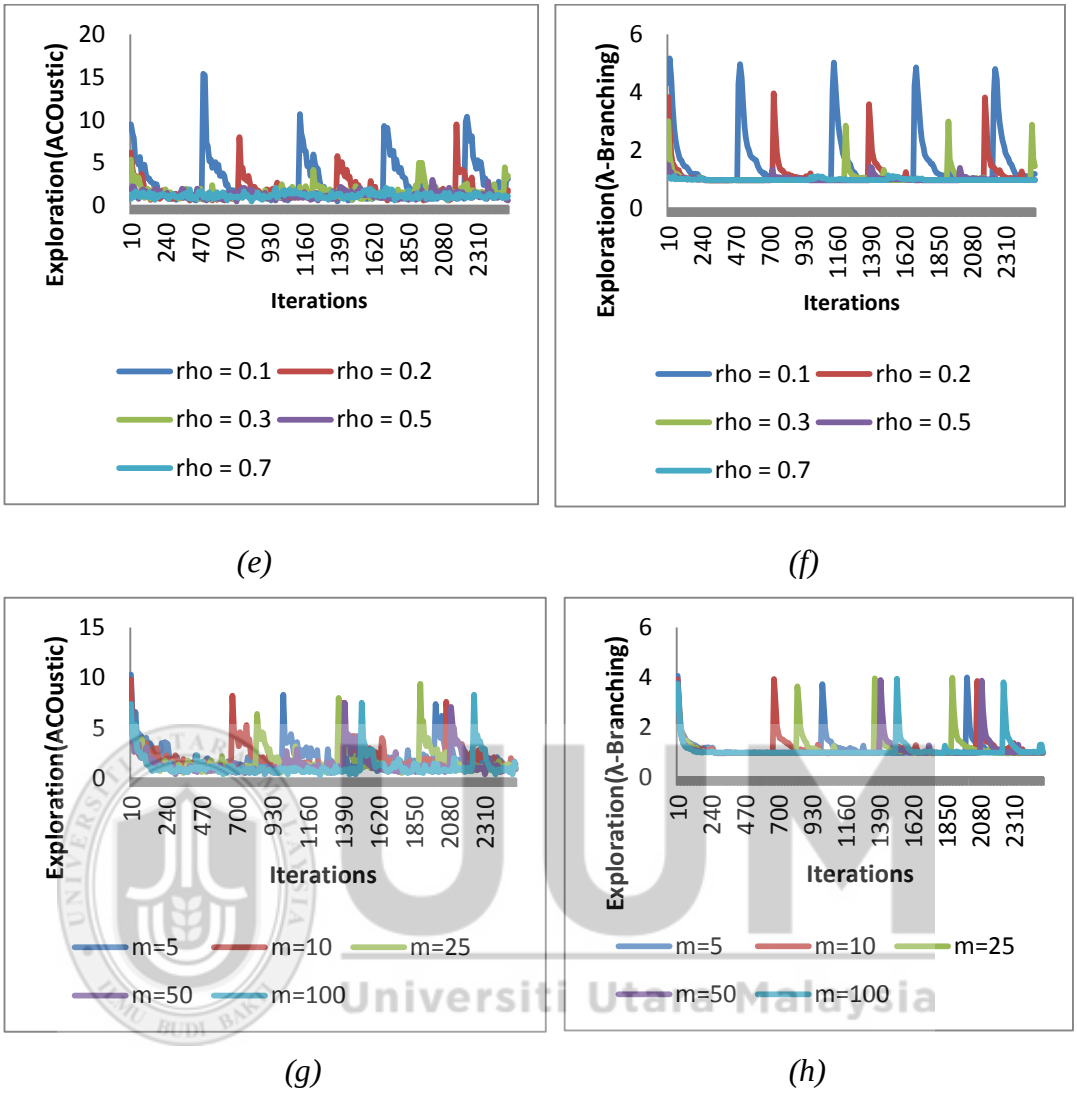


Figure 5.22. Results of Utilizing Acoustic (left) against Branching Factor (right) to Evaluate the Effect of pheromone intensity (alpha) (a-b), pre-heuristic effect (beta) (c-d), evaporation rate (rho) (e-f) and the number of ants (m)(g-h) in TSP

In Figure 5.22, the effect of varying the main parameters on the explorative and exploitative behavior of the MMAS *+rs +ls* algorithm is characterized. The proposed measure can detect the relationship between parameter values and local search. Figure 5.22 (a) shows clearly that the higher the value of pheromone intensity, the lower the exploration. The value ($\alpha = 1.0$) is the ideal value to achieve a moderate behavior. In Figure 5.22 (c), the influence of pre-heuristics about the instance of the

problem to be tackled is tested. The higher value of parameter (β), the more greedy behavior is recorded to become at its peak when (β = 10). The evaporation ratio as a trail learning factor has distinct impact on the way of search. Increasing the value of (rho) results in the slow learning of pheromone trail parameters, and thereby, the chance of forgetting the previous search experience will increase. In this way, the value of (rho = 0.7) leads to the increase of the exploitation. The value of (rho = 0.5) seems ideal (Figure 5.22 (e)). Figure 5.22 (g) reports that the exploration is decreasing with respect to the number of ants. The greater the number of ants, the lower the number of iterations performed in a run, consequently, the lower the number of different probability distributions used. A high value of m implies that the likely edges are often the same. These insights are compatible with the common believes among ACO researchers.

As shown, *ACO*ustic measure is a very convenient tool for characterizing the diversity of population. This conclusion came as a result of its robustness against the state-of-the art measures in ACO and of effectiveness as a statistical machine learning indicator. Besides, the role of the PbM scheme (presented in Chapter Four) is exist as the statistical and clustering information of *ACO*ustic is extracted from the population vector.

5.5 ACO-based Adaptive Parameters' Selection

The state-of-the-art methods for parameters' selection in ACO are self-adaptive methods (Pellegrini et al., 2012). They play a key role in solving several CO problems. However, they did not improve the performance of ACO in TSP and QAP,

except the work of Randall (2004) which has shown, explicitly, good results for TSP and QAP. There is an emphasis on adopting successful methodologies such as those in evolutionary metaheuristics.

In this thesis, the process of developing reactive method denoted as APS_{ACO} for solving the problem of parameters' selection in ACO adheres to the typical methodology in the field of parameters' selection of evolutionary algorithms. The process of proposing APS_{ACO} for solving the problem of parameters' selection in ACO is illustrated in Figure 5.23.

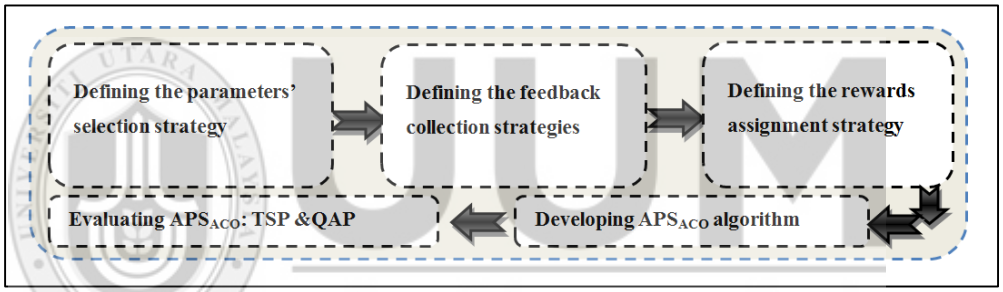


Figure 5.23. The Process of Developing the APS_{ACO}

Following the works of Failho (2010) and Aleti (2012), new insights for parameters' selection in ACO are obtained and motivate the proposal of APS_{ACO}. Table 5.1 shows the pros and cons of proposing APS_{ACO} compared with other adaptive parameters' selection methods.

The only disadvantage of the proposed method is its complex implementation. It is, for this context, similar to adaptive methods, while it differs in being free of the hyperparameters. It is able to adapt with the global and local characteristics of the

CO being tackled. These results from the robust indication for the exploration behavior, i.e. using *ACO*ustic, of the algorithm applied.

Table 5.1

Conceptual Comparison between APSACO and Other Adaptive Parameters' Selection Methods

Considerations	**Classical adaptive parameters' selection methods in ACO**			**Proposed APS_{ACO}**
	Pre-schedule	**Adaptive**	**Self-adaptive**	
Adapted with global characteristics	-	√	√	√
Adapted with local characteristics	-	√	-	√
Less augmented complexity	√	√	-	√
Less hyper parameters	√	-	√	√
Simple implementation	√	-	√	-
Algorithm structure independent	√	√	-	√
Follow a general methodology	-	-	√	√
Total scores	4	4	4	6

The most important feature of APS_{ACO} is its automatous search due to the independent pheromone matrix for parameters and values. Figure 5.24 depicts the general scheme of the proposed method.

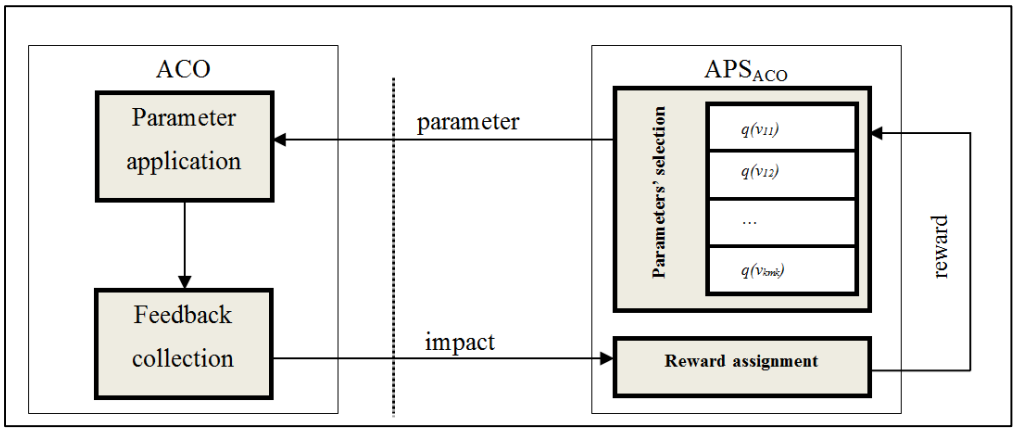


Figure 5.24. The General Scheme for APS_{ACO} Method

In Figure 5.24, the ACO algorithm asks APS_{ACO} which of the parameters can be applied for the current iteration/s. The *parameter selection strategy* selects a parameter according to its empirical quality (Section 5.6 includes further details) during the last iterations. The selected parameter is applied and its impact is transformed into a reward, using *reward assignment strategy*, to be used in updating the quality of parameters. The reinforcement learning process in APS_{ACO} is guided by those two strategies.

5.6 Parameters' Selection Strategy

In this strategy, the desirability of selecting the given parameter values *V* is affected proportionally by its empirical quality $Q = \{q(v_{11}), q(v_{12}), \ldots, q(v_{1m_1}), \ldots, q(v_{km_k})\}$ where *k* is the number of parameters and *m* is the number of values for the k^{th} parameter. Each parameter is associated with a range of values. The bounds of the ranges are set based on Dorigo and Stützle (2004). The j^{th} parameter value for the i^{th} parameter, i.e. the value v_{ij}, is selected as follows.

$$s(v_{ij}) = l_i + \frac{p(v_{ij})}{m}(u_i - l_i) \qquad 1 \leq i \leq k \qquad and \qquad 1 \leq j \leq m \tag{5.14}$$

$$p(v_{ij}) = q(v_{ij}) / \sum_{l=0}^{n} q(v_{il}) \tag{5.15}$$

where l_i is the lower bound value of i^{th} parameter; u_i is the upper bound value of i^{th} parameter; *m* is the number of values; and *p* is the proportional selection probability for value the v_{ij}. At the first application, the value of each parameter is chosen as the halfway point in its range. After that, the values are selected proportionally. If the

application is performed badly, then the application's desirability of the parameter has to be decreased, otherwise it will be rewarded. This is by firstly evaluating the effect of the selected values and then updating the empirical quality of the selected values.

5.7 Reward Assignment Strategies

In this strategy, the empirical quality of each parameter gains rewards only if the application of the parameter achieves impact for the optimization process. The impact determined by the feedback collection strategy is translated into rewards denoted as $r(v_{ij})$. The reward will be added to the previous quality of the perspective value as follows.

$$q(v_{ij}) = (1 - \rho) \cdot q(v_{ij}) + \rho \cdot r(v_{ij}) \quad (5.16)$$

The value of ρ is automatically assigned within the recommended range. Based on this formula, there is another instance of exploration versus exploitation: the best values are extensively used, while other values which need to be tried from time to time are not considered yet. In finding a good balance of the two processes, the bounding strategy is involved where the quantity τ_{min} represents the minimum selection probability for all the parameter values.

$$q(v_{ij}) = (1 - \gamma) \cdot q(v_{ij}) + \gamma \cdot \tau_{\min} \quad (5.17)$$

where the value of γ is automatically assigned by the proposed APS_{ACO} method itself.

Through this strategy, the effect of parameter value choices on the search is transformed into rewards. It involves the exploration state, the quality of solutions or both for rewards' calculation. To achieve this goal, three strategies are proposed, namely the *Quality-based Reward Assignment (QRA),* the *Exploration-based Reward Assignment (ERA),* and the *Unified Reward Assignment (URA)*.

5.7.1 Quality-based Reward Assignment

This QRA strategy relies on the improvement in the quality of solutions in assessing the effect of the current population. The median of the objective functions of the current population is used as an effect proxy for the application of selected parameter values. The value of rewards is calculated as follows.

$$r(v_{ij}) = {}^{1}/_{global\_avg} \tag{5.18}$$

The value of *global_avg* is the median of the objective function for the solutions that are recorded in the population-based memory.

5.7.2 Exploration-based Reward Assignment

In the ERA strategy, exploration is identified in terms of the relatedness amount between ants produced by *ACO*ustic as follows.

$$f(rltdnss) = \begin{cases} expl r = expl r + 1 & if\ rltdnss > X_{rltdnss} \quad (exploration) \\ expl = expl + 1 & otherwise \quad (exploitation) \end{cases} \tag{5.19}$$

where the value of $X_{rltdnss}$ is the first relatedness value captured when the number of clusters decrease. It is worth mentioning that this characterization function is

deactivated during the stagnation of the search. The stagnation is flagged when the solutions S_k since the last best restart i_{last} did not improve for the last ϵ iterations (e.g. 250 iterations), i.e. if $f(S_k) \geq f(S_{gb})$ and $i - i_{last} > \epsilon$. The rewards amount is derived from the impact of the application of parameter values which is calculated as follows.

$$r(v_{ij}) = \frac{explr^2}{expl} \tag{5.20}$$

It is worth mentioning that the values of the exploration/exploitation quantifiers, i.e. *explr* and *expl*, are very sensitive to the value of the nearest neighborhood threshold. The higher the threshold is, the more sensitive the quantifier becomes. In the beginning of the search, the amount of exploration starts higher than the exploitation one. With this property, the behavior of the algorithm is automated in various phases of the search. This automates the balance between exploration and exploitation in response to the current state of the search.

5.7.3 Unified Reward Assignment

The URA strategy relies on the quality of solutions and the diversity of solutions in assessing the effect of the current parameter values. The rewards are calculated as follows.

$$r(v_{ij}) = C_{num}/global\_avg \tag{5.21}$$

where the C_{num} is the number clusters. Based on this equation, the exploration behavior plays a fundamental role in determining the amount of rewards. The higher

the number of clusters is, the higher the reward becomes. The contributed strategies are emerged in the body of the ACO algorithmic framework as shown in Figure 5.25. In *initialization*, the probability and the quality vectors are initiated. In *select_param*, the parameter values are either selected as the halfway point in their ranges, if that is the first application, or selected proportionally to be involved in the search. The ants construct their solutions and update the memory of pheromone. The effect of the just applied parameter values is transformed into rewards by *assign_rewards*. It is based on the feedback collected from the search, and updates on the quality of the current parameter values. The amount of rewards assigned depends on the way of feedback collected whether it focuses on the improvement in quality, the improvement in exploration behavior, or the relative improvement in both of them.

Algorithm 5.5: APS$_{ACO}$

Set the number of parameters to k
Set the number of values to m
Set the maximum and minimum ranges of parameter values
Discretize the ranges R based on value of m
for $i = 1$ **to** k **do**
    $v_i \leftarrow r_i$
$d \leftarrow m/2$
for $i = 1$ **to** k **do**
for $j = 1$ **to** m **do**
    $q_{ij} \leftarrow \tau_{min}$ *// It can be set to τ_0 if another ACO variant is applied except the MMAS*
    $s_{ij} \leftarrow v_{id}$
while (not termination_condition()**) do**
    select_param ()
    construct_solutions ()
    update_ pheromone ()
    assign_rewards ()
end-while
end-algorithm

Figure 5.25. The Pseudocode of APS$_{ACO}$ Algorithm

5.8 Experimental Design for Developing APS_{ACO}

The goal of the empirical analysis is to evaluate the proposed APS_{ACO} against the state-of-the-art adaptation methods proposed for ant colony optimization. The implementation of four self-adaption methods is based on the work of two groups. The works of Randall (2004) and Forster et al. (2007) are in the first group, while the works of Martens et al., (2007) and Khichane et al. (2009) are in the second group. To capture the contribution of the groups independently of the problems or the algorithms for which they have proposed, the works of Randall and Khichane et al. are followed. In the first group, namely *RandallG*, the parameter values are selected online based on Randall's way (Randall, 2004), where the parameters are independent, e.g. the parameters β, ρ, γ and q_0. In the second group, namely *KhichaneG*, the parameter values are selected online based on Khichane's way (Khichane et al., 2009), where the parameters are interdependent, e.g. the parameters α and β. The search space for the parameter values must be known in advance and discretized in both groups, except in the second group where the values are optimized a priori. Both groups are in the same level at which they manage parameters. The rewards given to the parameter values selected during the run are based on the best-so-far ant in colony-level rather than the ant-level. The ant-level setting is omitted because most of the parameters are colony-wise, so that they cannot be adapted to multiple settings in each iteration. The number of parameter values *m* remains constant at 20. The ranges for the parameters q_0, ρ and γ are bound between the constant values of 0 and 1; for the parameter β is bound between the constant values of 5 and 1 and for the parameter α is bound between 1 and 2.

The MMAS algorithm is involved as a test-bed. It is used for solving TSP and QAP. For solving TSP, the MMAS is included in ACOTSP.V1.3 software (Stützle, 2004). The implementation of MMAS for QAP is based on the work of Stützle and Hoos (2000) and follows the general algorithmic framework of ACOTSP.V1.3. The pseudo-random proportional rule that is used in ACS is used in MMAS as well to gain high performance by default. A 2-opt local search procedure is used with MMAS for all QAP instances.

The experiments are conducted on a Windows 8 64-bit operating system, processor Intel Core i3-3217U with CPU @ 1.80GHz, RAM 4GB. Each experiment is executed ten times to avoid the stochastic behavior. A maximum of 10 seconds is used as a termination condition for the run of particular algorithms. The QAP and TSP instances are selected from the QAPLIB and TSPLIB repositories as in Table 5.2.

Table 5.2

The TSP and QAP Instances used in the Evaluation

TSP			**QAP**		
Name	**Size**	**Best-Known Cost**	**Name**	**Size**	**Best-Known Cost**
eil51	51	426	nug15	15	1150
st70	70	675	nug20	20	2570
eil76	76	538	tai25a	25	1167256
gr96	96	55209	tai35a	35	2422002
rd100	100	7910	ste36a	36	9526
bier127	127	118282	tho40	40	240516
d198	198	15780	sko49	49	23386

5.9 Results of APS$_{ACO}$'s Application

The adaptive parameters' selection method has been applied to automate the exploration and exploitation during the run. In order to report the results of

application, non-parametric descriptive statistics are used. This is because the distribution of results is non-normal. The cost results are reported using RPD as a comparative measure. In order to track the RPD values at any time, CTM test has been utilized. Table 5.3 reports the results of the conducted experiments in which APS_{ACO} algorithm succeeded in accelerating the convergence to high quality solutions produced by the ACO algorithm. At the first iterations, APS_{ACO} assigns initial values to the perspective ACO parameters. Next, the values are selected proportionally according the quality of solutions. The desirability of selecting one parameter relies on its empirical quality (see Equation 5.16). Since the parameters of ACO algorithm are the main player in adjusting exploration and exploitation, their values will be selected automatically. This automation entails the direct projection on the exploration/exploitation behavior.

In Table 5.3, the results reveal that APS_{ACO} with QRA shows very good results on QAP and less in TSP compared with other state-of-the-art methods. In TSP, while occasionally *RandallG* finds the best quality solutions (such as st70, gr96 and d198), the overall behavior of APS_{ACO} with QRA is better for small TSP problems without local search. In QAP, *KhichaneG* sometimes finds the best solution (such as tai35a). However, the overall performance of the proposed method is better.

In Table 5.4, the comparison of the three proposed strategies of reward assignment is depicted. The QRA strategy was the best performance, while the ERA showed less performance and the URA came in the last.

Table 5.3

The Results of Evaluating APS_{ACO} (QRA) against Other Parameters' Selections Methods in TSP and QAP using RPD Test

TSP/QAP Problem	RandallG				KhichaneG				APS_{ACO} (QRA)			
	cost			runtime	cost			runtime	cost			runtime
	min	med	max	med	min	med	max	med	min	med	max	med
eil51	0.23	0.77	2.11	1.80	0.46	0.86	1.17	4.40	**0.0**	**0.11**	**0.46**	**2.41**
st70	**0.14**	1.51	3.25	6.52	0.74	2.28	4.14	4.26	0.44	**0.91**	**2.37**	**6.49**
eil76	0.18	1.41	2.97	3.77	0.92	1.89	2.60	4.86	**0.0**	**0.29**	**0.92**	5.70
gr96	**0.33**	1.39	4.68	4.22	1.91	3.08	4.76	5.61	0.57	**1.13**	**1.55**	4.73
rd100	0.05	1.10	3.53	5.27	1.01	2.64	6.11	6.91	0.32	**0.76**	**1.87**	8.24
bier127	1.61	2.96	5.24	8.20	2.80	3.76	4.98	3.39	**1.53**	**2.48**	**3.61**	4.99
d198	**1.96**	4.21	6.82	7.74	3.07	4.96	6.84	8.89	2.20	**3.94**	**5.48**	**6.14**
nug15	0.0	0.0	0.0	0.22	0.0	0.0	0.0	0.19	0.0	0.0	0.0	**0.05**
nug20	0.0	0.04	0.15	2.75	0.0	0.01	0.15	2.64	0.0	**0.0**	**0.0**	**1.76**
tai25a	1.4	2.2	2.7	4.77	1.68	2.24	2.61	5.54	**1.21**	**1.88**	**2.55**	**4.50**
tai35a	2.98	3.22	3.6	4.5	**2.58**	3.12	3.51	6.13	2.61	**3.02**	**3.38**	**3.77**
ste36a	2.45	3.3	4.2	6.14	1.61	3.14	4.42	4.13	**0.92**	**2.26**	**3.19**	**3.71**
tho40	1.52	1.88	2.11	5.31	1.39	1.98	2.32	6.33	**1.05**	**1.73**	**2.08**	**4.92**
sko49	1.08	1.38	1.75	6.06	0.95	1.32	1.54	**4.51**	**0.85**	**1.18**	**1.40**	5.44

Table 5.4

The Results of Evaluating APS_{ACO} using QRA, URA and ERA in TSP and QAP using RPD Test

TSP/QAP Problem	APS_{ACO} (QRA)				APS_{ACO} (URA)				APS_{ACO} (ERA)			
	cost			runtime	cost			runtime	cost			runtime
	min	med	max	med	min	med	max	med	min	med	max	med
eil51	0.0	**0.11**	**0.46**	**2.41**	0.23	0.37	0.7	4.94	0.0	0.28	1.17	2.63
st70	0.44	0.91	2.37	6.49	0.44	1.20	3.25	4.35	**0.29**	**0.90**	**1.92**	**5.80**
eil76	0.0	**0.29**	0.92	5.70	0.0	0.44	1.11	5.70	0.0	0.61	0.92	4.65
gr96	0.57	1.13	**1.55**	4.73	**0.36**	1.03	3.28	5.01	0.38	**0.98**	2.82	8.12
rd100	0.32	0.76	1.87	8.24	0.05	1.18	2.98	5.91	**0.01**	**0.13**	**0.91**	6.63
bier127	**1.53**	**2.48**	**3.61**	**4.99**	1.61	2.75	4.57	6.35	2.14	3.20	4.22	6.08
d198	2.20	3.94	5.48	6.14	2.74	4.40	6.36	7.79	5.15	8.79	7.43	3.51
nug15	0.0	0.0	0.0	**0.05**	0.0	0.0	0.0	0.22	0.0	0.0	0.0	0.27
nug20	0.0	**0.0**	0.0	**1.76**	0.0	0.04	0.15	3.93	0.0	0.01	0.15	3.52
tai25a	1.21	**1.88**	2.55	4.50	1.21	2.01	**2.42**	3.67	1.3	1.9	2.3	5.7
tai35a	2.61	**3.02**	**3.38**	**3.77**	3.13	3.34	3.72	5.09	**2.6**	3.10	3.4	5.36
ste36a	**0.92**	**2.26**	**3.19**	**3.71**	2.09	3.0	3.82	5.96	1.4	2.9	4.0	4.11
tho40	1.05	**1.73**	**2.08**	**4.92**	1.26	1.77	2.15	5.45	**0.9**	1.84	2.3	5.97
sko49	**0.85**	**1.18**	**1.40**	**5.44**	1.06	1.39	1.71	5.54	1.23	1.41	1.63	4.67

The design of each of the proposed strategies determines the suitable situation to apply any of them. For example, when a restart mechanism is applied, the URA will be the promising choice because of its tendency to increase current exploration.

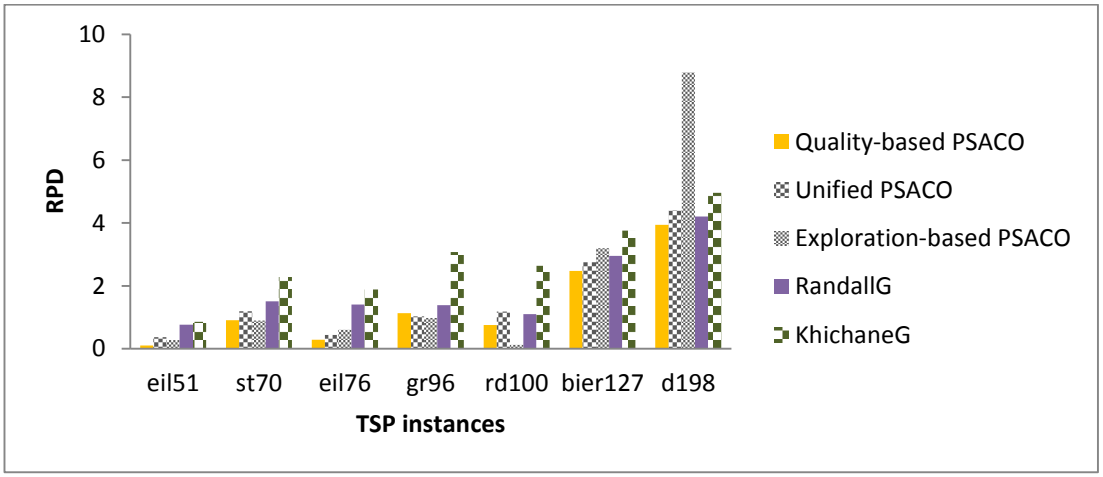


Figure 5.26. The Results of Evaluating APS_{ACO} using QRA, URA and ERA against Other Parameters' Selections Methods in TSP using RPD Test

In Figure 5.26, the overall performance of the proposed strategies outperforms the state-of-the-art methods for TSP instances. With small size instances, the three methods are the best. The QRA strategy was the best among all. In some cases, the ERA strategy outperforms (such as gr96 and rd100). However, when the size of the problem increases, the ERA has a worse performance because of its additional computations. The URA strategy is more robust.

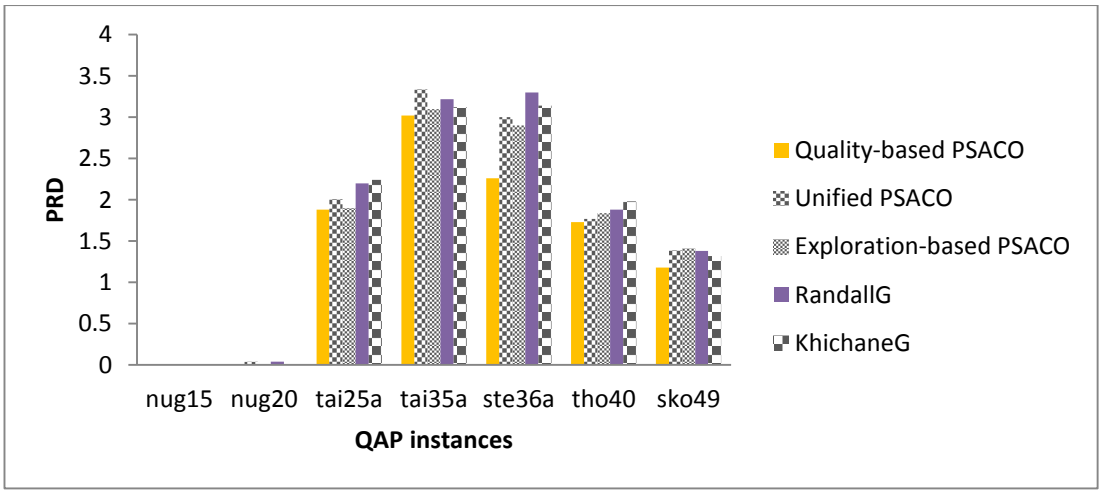


Figure 5.27. The Results of Evaluating APS_{ACO} using QRA, URA and ERA against Other Parameters' Selections Methods in QAP using RPD Test

In Figure 5.27, the overall performance of the proposed strategies outperforms the state-of-the-art methods for QAP instances. With small size instances, the URA does not much improve the quality of solution comparing with QRA, ERA and KhichaneG. The QRA strategy produced the best average quality of solutions in all experiments.

5.10 Summary

The exploration measurement and the adaptive parameters' selection in ACO are discussed in this chapter. For the exploration measurement, the problem of robustness in machine learning-based indicators has been solved by emerging simple statistics about current exploration to the design of those indicators. The results illustrated that the proposed indicator, denoted by *ACO*ustic as inspired from the acoustic mimicry in nature, is more informative and more robust.

For the adaptive parameters' selection, the general schema of parameter adaptation is adopted based on successful methodologies in the field of evolutionary algorithms. Two independent issues are highlighted in the schema: the parameters' selection strategy and the reward assignment strategy. Four parameter adaptation algorithms are implemented into two groups: RandallG and KhichaneG implementations. For the parameters' selection, the implementation of the first group is followed. For the reward assignment, three contributed strategies, denoted by QRA, ERA and URQ, are proposed for improving the performance of the existing parameter adaptation methods in ACO. Three variants of APS_{ACO} algorithm are produced by varying the proposal of the perspective strategy. In the design of QRA, the general improvement

in the quality of solutions used a proxy for the impact of the selected parameters, while in ERA and URQ, the feedback collected via *ACO*ustic is alternately the first proxy. By the said contribution, the parameters' selection problem in ACO is addressed. The effectiveness of the proposed APS_{ACO} variants is evaluated against each other and against the state-of-the-art methods. Results showed that APS_{ACO} with QRA is the best among all. As the automation of exploration and exploitation has been implemented, Chapter 6 discusses projection of the proposed exploration and exploitation components on top of RMMAS algorithm to develop a more advanced reactive approach.

CHAPTER SIX
PROPOSED REACTIVE APPROACH FOR AUTOMATING EXPLORATION AND EXPLOITATION IN ACO

6.1 Introduction

The development of the proposed reactive ant colony optimization approach is presented. This chapter summarizes the final performance of RACO. Through the previous chapters, several exploration and exploitation components have been proposed and tested separately, whereas in this chapter, the components are merged. Section 6.2 depicts the general scheme of the proposed approach, namely RACO. Section 6.3 presents the experimental design of the RACO evaluation. The results and analysis for TSP and QAP are presented in Sections 6.4 and 6.5 respectively. The summary of the chapter is presented in Section 6.6.

6.2 Proposed Reactive Approach

The general scheme of RACO is presented in Figure 6.1. RACO starts solving CO problems by iterating two activities, namely ants' activity and queen's activity. An example of the ants' activity is the probabilistic solution construction where each ant is able to take individual decisions. An example of the queen's activity is every central decision can be taken to change the current search status. CO problems (such as TSP and QAP) are assembled as a finite set of solution components. Next, a set of pheromone values called the pheromone model is defined. The set of pheromone values is parameterized probabilistically to be used then in generating solutions

based on the solution components. Two reactive memory schemes, CbM and PbM, are defined in Chapter Four. CbM is derived from the solution components, while PbM is derived from the overall population of solutions. The candidate solutions are constructed using the pheromone model. The pheromone values are updated by the queen in such a way that it is biased in future towards high quality solutions.

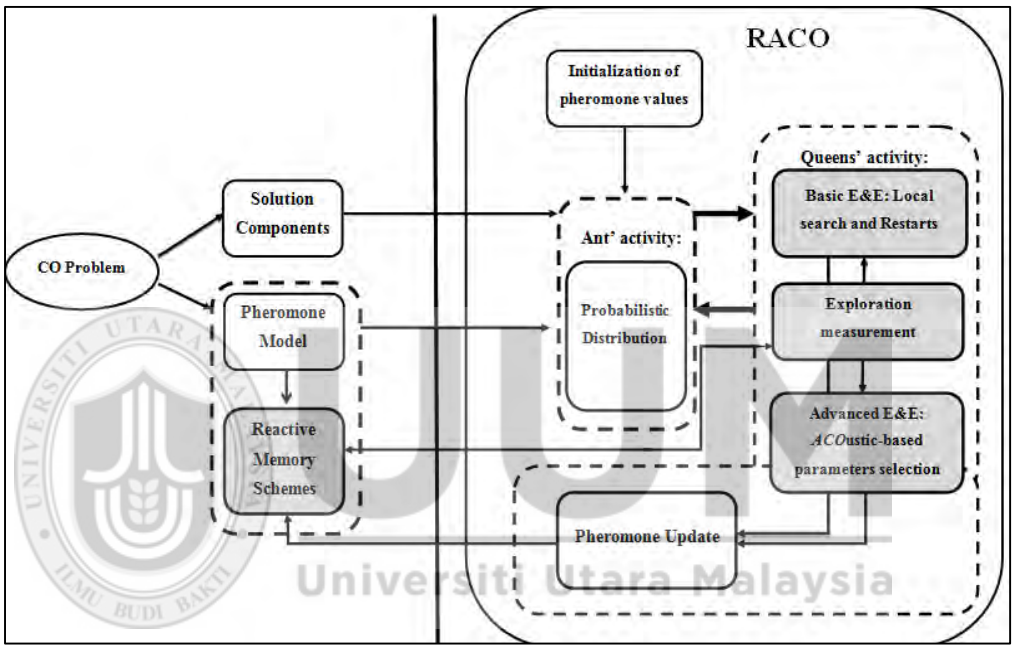


Figure 6.1. The General Scheme of RACO

There are two different neighborhood structures; one is framed by the local search procedures, and the other one is framed by the ants. For this part of RACO, there are two basic E&E mechanisms: the reactive restart mechanism and the RLS mechanism. In the former mechanism, the neighborhood drawn by ants is traversed before the restart just to record the unpromising regions. The regions are simply characterized using τ_{min} threshold where the components of solutions below this threshold will be recorded in the CbM scheme in terms of reactive heuristics. After

restart, the reactive heuristic will be used as guidance for ants to decide the next component in the constructed solution. A high priority of selections is given to components associated to the reactive heuristics. In the latter mechanism, i.e. RLS, relies on the neighborhood structure drawn by local search procedures. An *old-best-so-far* solution will be recorded in the PbM scheme to be used in future iterations as a reference for improvement in the quality of solutions. If the new produced solution is better than the *old-best-so-far* solution, it will be recorded in the memory; otherwise, the *old-best-so-far* solution will be recorded again in the memory.

The second part of the queen's activity is the exploration measurement. Using several exploration measures and absolute triggers, the queen characterizes the current state of search whether it is exploration or exploitation, then promotes a suitable reaction. The queen in this way controls the reinforcement learning process inside the colony by forcing other ants for being exploitative agents or being explorative ones. In the former choice, they keep searching around the structure of the neighborhood of good solutions, whereas, in the latter choice, they shift the search to another neighborhood structure. Several exploration indicators used within RACO in order to redirect the current search state from exploitation to exploration directly using the reactive restart mechanism. Traditional exploration indicators (such as λ-branching factor and acceptance criteria) are involved. A machine learning mechanism of indication called *ACO*ustic (detailed in Chapter Five) emerges in this part. Using this indication mechanism, the exploration and exploitation can be absolutely quantified, put in relation with the quality of solutions in a unified way or relied on the quality of solutions only in a relative way.

The last part of the queen's activities is the APS_{ACO} mechanism, in which the queen controls the way of the search based on the feedback collected from the search process. The mechanism automates the exploration and exploitation during the run based on the feedback collected. Internal reinforcement learning process is involved to learn the parameter values during the run. Through two strategies of parameters' selection and reward assignment, the process is maintained. In the first iteration, the values are selected from the midway of their perspective ranges. Hereafter, they are selected in a proportional way to their approximate effect on the optimization process. The effect transforms into a reward to be assigned to the parameter values producing good quality solutions, good exploration/exploitation behavior, or good balance between the both of them.

6.3 Experimental Design for RACO Evaluation

The performance of RACO is evaluated by the comparison with other metaheuristics approaches to solve TSP and QAP. The evaluation metric is reported using the RPD test. The maximum number of iterations is equal to the same number of tours for the algorithms with which RACO is compared. An average of ten trails for the results is reported. For RACO parameter settings, the neighborhood threshold is fixed to (0.8) without tuning. The number of ants (m) is equal to (5), while the rest of the RACO parameters are configured adaptively using the ERA strategy. Hence, the RACO variant used in the experiments is denoted as $RACO_{ERA}$.

For TSP, the instances are taken from TSPLIB (Reinelt, 1991), and then categorized into small, medium, and large sizes. *Burma14, Dantzig42, Oliver30, Eil51, Eil76,*

KroA100 and Eil101 are categorized as small, *d198, lin318, att532 and rat783* as medium, and *pr1002, u1060, pcb1173, d1291 and fl1577* as large sizes. The configuration of experiments is dictated based on the availability of the published results. Numerical experiments are executed to regenerate the results of other algorithms; otherwise, their performance is taken from the literature. The results of ACS and six MMAS variants are based on the implementation included in ACOTSP.V1.3 (Stützle, 2004), while the results of the iterated local search (ILS) are from Stützle (1998). Other algorithms with which RACO is compared are simulated annealing (SA), evolutionary programming (EP), genetic algorithm (GA), particle swarm optimization (PSO), and artificial bee colony (ABC). The results of SA and EP are from Dorigo and Gambardella (1997). The results of GA and PSO are from Çunkaş and Özsağlam (2009) and the results of ABC are from Kocer and Akca (2014).

For QAP, the benchmarking data are taken from QAPLIB (Burkard et al., 1997), and then classified into real-life, real-life-like and random-generated categories. These are *bur26a, bur26b, bur26c, bur26d, bur26e, bur26f, bur26g, bur26h, chr25a, els19, kra30a and kra30b* for the real-life category, *tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b and tai80b* for the real-life-like category, and *Nug30, Ste36b, Tai30a, Tai40a, Tai50a, Tai60a, Tai80a and Tai100a* for the random-generated category. The configurations of long-run and short-run are conducted on real-life and real-life-like instances. Different sizes of instances are tackled. The results of the algorithms used in the comparison are taken from the literature. The performance of MMAS, robust tabu search (Ro-TS), reactive tabu search (RTS), SA, genetic hybrid (GH),

and hybrid ant system (HAS-QAP) are from Stützle and Hoos (2000) and Stützle (1999), while the performance of object-guided ant colony optimization (OG-ACO) and hybrid artificial fish-school optimization (HAFSO) are from Ziqiang and Yi (2014). The run length is dynamic and is based on Gambardell, Thallard and Dorigo (1997) for the real-life and real-life-like instances, while for the random-generated instances; it is based on Ziqiang and Yi (2014).

6.4 Results of the TSP Experiments

RACO is applied to small-, medium- and large-sized TSP instances as shown in Tables 6.1, 6.2 and 6.3. In the small-sized instances' experiments, the proposed algorithm achieved a 100% success rate by reaching the known optimum at the first four turns. The rate was 99% and 92% for the fifth and sixth turns. It was observed that RACO ended with 0% margin of error in small-sized TSP problems. The results confirmed that the combination of RH heuristics, RLS technique and QRA controller produce high quality solutions.

In the medium-sized instances of experiments, the obtained RPT rates in the medium size were competitive. As it can be clearly seen in Table 6.2, the solutions of test problems d198, lin318, pcb442, att532 and rat783 through RACO produced the smaller error rate than others, except in MMAS-w algorithm, which was similar with lin318 and rat 783. Sometimes, the difference in performance is not very noticeable (such as d198 and lin318) between RACO and MMAS variants, but RACO monopolized its outperformance in all turns. In addition, the competitive results were produced by different MMAS variants and this is not the case with RACO. Some

TSP instances (such as lin318) contain several local minima, and to escape from those, the algorithm must change its behavior accordingly and in an online manner, which is the case with RACO. In general, this result gives an indicator that the quality of solutions produced by RACO is better due to the impedance of its search to be stagnated.

Table 6.1

Results of Comparing RACO with ACS, EP, SA, GA, PSO and ABC Algorithms in Small Size TSP Instances using RPD Test

TSP instance	*ACS*	*EP*	*SA*	*GA*	*PSO*	*ABC*	*RACO*
Burma14	-	-	-	0.92	1.14	0.33	**0.00**
Oliver30	**0.00**	**0.00**	0.95	-	-	-	**0.00**
Dantzig42	-	-	-	2.05	2.08	0.71	**0.00**
Eil51	0.51	0.24	4.24	2.11	2.45	1.89	**0.09**
Eil76	1.71	1.31	8.41	2.56	3.16	6.31	**0.13**
KroA100	1.17	-	-	2.68	3.71	2.16	**0.82**

Table 6.2

Results of Comparing RACO with MMAS Variants, ACS+3-opt and ILS+3-opt Algorithms in Medium Size TSP Instances using RPD Test

TSP instance	*MMAS-w*	*MMAS-wnh*	*MMAS-wnts*	*MMAS-t*	*MMAS-tnh*	*ACS+3-opt*	*ILS-3-opt*	*RACO*
d198	0.003	0.002	0.003	**0.000**	0.002	0.002	0.002	**0.000**
lin318	**0.000**	0.011	0.078	0.005	0.02	0.17	0.085	**0.000**
pcb442	0.26	0.25	0.24	0.25	0.24	0.26	0.28	**0.010**
att532	0.079	0.091	0.082	0.15	0.096	0.17	0.086	**0.040**
rat783	**0.095**	0.14	0.12	0.20	0.22	0.28	0.34	**0.095**

In the large-sized instances of experiments, the results reported in Table 6.3 can be used to evaluate the scalability in addition to the quality of solutions because of the vast landscape of the tested TSP instances. Only MMAS-w and MMAS-wrnt were

considered in this part of comparison. It is observed that RACO performs well in terms of solution qualities and naturally in scalability because of the utilization of long-term exploration and exploitation dynamics. With large sized instances, the importance of reactive heuristics becomes less because the lower number of restart triggers. This observation suggests exchanging those traditional triggers (such as λ-branching factor) with machine learning triggers (such as *ACO*ustic).

Table 6.3

Results of Comparing RACO with MMAS Variants, ACS+3-opt and ILS+3-opt Algorithms in Large Size TSP Instances using RPD Test

TSP instance	***MMAS-w***	***MMAS-wnts***	***ACS***	***ILS-3-opt***	***RACO***
pr1002	0.30	0.18	0.41	0.21	**0.14**
u1060	0.34	0.36	0.29	**0.14**	0.26
pcb1173	0.11	0.095	0.37	0.24	**0.001**
d1291	0.041	0.055	0.14	0.15	**0.037**
fl1577	0.28	**0.10**	0.35	0.65	0.022

6.5 Results of the QAP Experiments

RACO is applied to real-life, real-life-like and random-generated QAP instances as shown in Tables 6.5-6.9. The results of the experiments on real-life and real-life-like instances are reported for short and long runs. The results confirmed that the quality of solution produced by the RACO algorithm is better than others for QAP.

From Table 6.5, where the RACO behavior is under strong time constraints, it is clear that RACO is well adapted to the real-life instances of QAP. For the *bur26x* instance, the results can show that the population-based methods (such as RACO, GH and HAS-QAP) perform better than the local search-based (SA and Ro-TS). In fact, Ro-TS and SA are not really competitive for these kinds of problems. For the

els19 instance, the ant-based metaheuristics are the better. For *kra30a*, HAS-QAP, GA and SA seem to be the best methods, whereas RACO produced the worse solutions. It is known that the performance of competitive algorithms rely on the type of QAP problems.

Table 6.5

Results of Comparing RACO with Ro-TS, RTS, SA, GH, HAS-QAP and MMAS-QAP$_{3\text{-}opt}$ Algorithms in Real-Life QAP Instances for Short Run using RPD Test

QAP instance	***Best-Known Solution***	***Ro-TS***	***RTS***	***SA***	***GH***	***HAS-QAP***	***MMAS-QAP$_{3\text{-}opt}$***	***RACO***
real-life instances:								
bur26a	5426670	0.208	—	0.185	0.060	0.027	**0.010**	**0.000**
bur26b	3817852	0.441	—	0.191	0.090	0.106	**0.000**	**0.000**
bur26c	5426795	0.170	—	0.137	0.004	0.009	**0.000**	**0.000**
bur26d	3821225	0.249	—	0.379	0.003	0.002	**0.000**	**0.000**
bur26e	5386879	0.076	—	0.228	0.003	0.004	**0.000**	**0.000**
bur26f	3782044	0.369	—	0.224	0.006	**0.000**	**0.000**	**0.000**
bur26g	10117172	0.078	—	0.139	0.006	**0.000**	**0.000**	**0.000**
bur26h	7098658	0.349	—	0.368	0.003	0.001	**0.000**	**0.000**
chr25a	3796	15.969	16.844	27.139	15.158	15.690	20.18	**9.53**
els19	17212548	21.261	6.714	16.028	0.515	0.923	0.170	**0.000**
kra30a	88900	2.666	2.155	1.813	1.576	**1.664**	7.551	6.068
kra30b	91420	0.478	1.061	1.065	0.451	0.504	0.964	**0.180**

From Table 6.6, where the run is longer, the results obtained with RACO were competitive to other ant-based algorithms for some instances, while it was better for most of the instances. For the *bur26x* instance, all the runs of RACO, MMAS-QAP$_{3\text{-}opt}$ and HAS-QAP succeeded in finding the best solution known. The behavior of RACO and HAS-QAP is equivalent, except that RACO produced the best solution for the *kra30b* instance, whereas HAS-QAP was the best in solving the chr25a

instance. For the kra30a instance, the iterative methods such as GH are the best, while RACO and MMAS-QAP$_{3\text{-opt}}$ are the worst. From this table, the same conclusions for shorter runs can be drawn. In general, the behavior of RACO relies on the shape of the fitness landscape of QAP.

Table 6.6

Results of Comparing RACO with Ro-TS, RTS, SA, GH, HAS-QAP and MMAS-QAP$_{3\text{-opt}}$ Algorithms in Real-Life QAP Instances for Long Run using RPD Test

QAP instance	***Best-Known Solution***	***Ro-TS***	***RTS***	***SA***	***GH***	***HAS-QAP***	***MMAS-QAP$_{3\text{-opt}}$***	***RACO***
real-life instances:								
bur26a	5426670	0.0004	—	0.1411	0.0120	**0.000**	**0.000**	**0.000**
bur26b	3817852	0.0032	—	0.1828	0.0219	**0.000**	**0.000**	**0.000**
bur26c	5426795	0.0004	—	0.0742	**0.000**	**0.000**	**0.000**	**0.000**
bur26d	3821225	0.0015	—	0.0056	0.0002	**0.000**	**0.000**	**0.000**
bur26e	5386879	**0.000**	—	0.1238	**0.000**	**0.000**	**0.000**	**0.000**
bur26f	3782044	0.0007	—	0.1579	**0.000**	**0.000**	**0.000**	**0.000**
bur26g	10117172	0.0003	—	0.1688	**0.000**	**0.000**	**0.000**	**0.000**
bur26h	7098658	0.0027	—	0.1268	0.0003	**0.000**	**0.000**	**0.000**
chr25a	3796	6.9652	9.8894	12.4973	2.6923	**3.0822**	9.43	7.48
els19	17212548	**0.000**	0.0899	18.5385	**0.000**	**0.000**	**0.000**	**0.000**
kra30a	88900	0.4702	2.0079	1.4657	**0.1338**	0.6299	6.40	7.01
kra30b	91420	0.0591	0.7121	0.1947	0.0536	0.0711	0.11	**0.020**

The results of the experiments on the real-life-like instances are reported in Tables 6.7 and 6.8. For short runs, the results showed that RACO has succeeded in finding the best solutions for all instances. It can be concluded from the short run experiments that RACO produces high quality solutions earlier than other methods. Therefore, it is suitable to deal with anytime applications. For long runs, the results confirm the robustness of the proposed approach. On the other side, other

population-based methods (such as GH, HAS-QAP and MMAS-QAP$_{3\text{-opt}}$) showed better behavior than local-search methods (such as Ro-TS and SA) for the instances of *taixxb* type. The reason behind the outperformance of the population-based algorithms is that they were of higher exploration. Yet, the role of reactive heuristics dominates the role of other exploration components of RACO.

Table 6.7

Results of Comparing RACO with Ro-TS, RTS, SA, GH, HAS-QAP and MMAS-QAP$_{3\text{-opt}}$ Algorithms in Real-Life like QAP Instances for Short Run using RPD Test

QAP instance	***Best-Known Solution***	***Ro-TS***	***RTS***	***SA***	***GH***	***HAS-QAP***	***MMAS-QAP$_{3\text{-opt}}$***	***RACO***
real-life like instances:								
tai20b	122455319	6.700	—	14.392	0.150	0.243	0.170	**0.000**
tai25b	344355646	11.486	—	8.831	0.874	0.133	0.316	**0.006**
tai30b	637117113	13.284	—	13.515	0.952	0.260	0.262	**0.001**
tai35b	283315445	10.165	—	6.935	1.084	0.343	0.591	**0.040**
tai40b	637250948	9.612	—	5.430	1.621	0.280	0.382	**0.004**
tai50b	458821517	7.602	—	4.351	1.397	0.291	0.545	**0.146**
tai60b	608215054	8.692	—	3.678	2.005	0.313	0.673	**0.136**
tai80b	818415043	6.008	—	2.793	2.643	1.108	1.292	**0.592**

The results of the experiments on the random-generated instances show that RACO is better than other methods with all the scales of this type of QAP instances. This superiority to the modern swarm intelligence methods, i.e. OG-ACO and HAFSOA, confirms the harmony in combining the exploration and exploitation components of RACO.

Table 6.8

Results of Comparing RACO with Ro-TS, RTS, SA, GH, HAS-QAP and MMAS-QAP$_{3\text{-}opt}$ Algorithms in Real-Life like QAP Instances for Long Run using RPD Test

QAP instance	*Best-Known Solution*	*Ro-TS*	*RTS*	*SA*	*GH*	*HAS-QAP*	*MMAS-QAP$_{3\text{-}opt}$*	*RACO*
real-life like instances:								
tai20b	122455319	**0.000**	—	6.7298	**0.000**	0.0905	**0.000**	**0.000**
tai25b	344355646	0.0072	—	1.1215	**0.000**	**0.000**	0.04	**0.000**
tai30b	637117113	0.0547	—	4.4075	0.0003	**0.000**	0.08	**0.000**
tai35b	283315445	0.1777	—	3.1746	0.1067	0.0256	0.32	**0.061**
tai40b	637250948	0.2082	—	4.5646	0.2109	**0.000**	0.14	**0.000**
tai50b	458821517	0.2943	—	0.8107	0.2142	0.1916	0.30	**0.093**
tai60b	608215054	0.3904	—	2.1373	0.2905	0.0483	0.36	**0.049**

Table 6.9

Results of Comparing RACO with OG-ACO and HAFSOA Algorithms in Random Generated QAP Instances using RPD Test

QAP instance	*Best-Known Solution*	*OG-ACO*	*HAFSOA*	*RACO*	*Number of Iterations*
random generated instances:					
Nug30	6124	0.294	0.291	**0.007**	1500
Ste36b	15852	1.336	0.804	**0.000**	1800
Tai30a	1818146	1.864	1.772	**1.567**	1500
Tai40a	3139370	2.597	2.306	**1.943**	2000
Tai50a	4941410	2.934	2.685	**2.328**	2500
Tai60a	7208572	2.904	2.669	**2.533**	3000
Tai80a	13557864	2.666	2.169	**2.081**	4000
Tai100a	21125314	2.517	2.233	**2.035**	5000

6.6 Summary

The concern that the E&E components of RACO: reactive heuristics, recursive local search, and the quality-based reward assignment within the APS$_{ACO}$ parameter controller, may interdependently conflict each other when they run together, are refuted. The schema of how the components interconnect has been figured in this

chapter. Results showed that the said components are operating harmoniously as RACO has superior to sixteen metaheuristic algorithms. The generality of RACO as an effective method for combinatorial optimization enables further extensions as explained in the next chapter.

CHAPTER SEVEN
CONCLUSION AND FUTURE WORK

Ant colony optimization metaheuristic solves, stochastically, optimization problems by transforming a biological approach of real ants for finding food into a computational approach for finding high quality solutions. Traversing the search space of the problem challenged by a dilemma called the exploration versus the exploitation. Many previous studies have addressed important aspects of the dilemma, such as the role of memory models in learning while optimizing, the way in which those models are managed, e.g. pheromone model management, and the parameter setting. The problem with those studies is that they tried to solve the exploration and exploitation problem by focusing on one of the perspectives while neglecting the others as concluded in Chapter Two.

Reactive search is a technique to improve the internal behavior metaheuristics by automating the exploration and exploitation states of search in online and offline manners. The feedback collected during the search reports to the user, in offline approaches, or reports to the algorithm itself, in online approaches, to evaluate the current state of the search and performs the suitable reaction to adjust it. Unfortunately, in reactive-based ACO methods, the online approaches are premature compared with offline ones. There are no general guidelines for adopting reactive search in improving the exploration and exploitation balance within ACO. Chapter Three proposes a unified methodology for improving the three aspects of reactive-based ACO: memory, exploration indication, i.e. feedback, and parameterization.

7.1 Research Contributions

This thesis presented a new ACO-based reactive approach for automating exploration and exploitation of ant colony optimization algorithms during the optimization process. The main contributions of the thesis are new reactive-based memory models, new nature-inspired exploration indicator, and new adaptive parameter selection strategy.

In the first contribution, two memory-based components are provided: reactive heuristics (RHs) and recursive local search (RLS) (see Subsections 4.2.2 and 4.2.6) respectively. The arcs that their pheromone amount became below a predefined threshold are recorded in the CbM scheme in terms of reactive heuristics. RHs are deactivated until some events trigger them such as the occurrence of stagnation. The trigger activates the use of these heuristics after restarting the current search. The proposed heuristics improved the behavior of the restart mechanism and produced good results. Next, the problem of premature exploitation has addressed by the proposal of RLS, the exploitation mechanism. RLS records a population of solutions instead of arcs using the PbM scheme, which is a fixed size vector of the high quality solutions found in current and previous iterations. The imperial results showed that solving small and medium TSP instances are more profitable from RHs than larger instances. For all sizes of QAP instances, the RLS mechanism gives a higher impact than applying it to TSP. Two variants of MMAS, they are RMMAS and $RMMAS_{RLS}$ (refer to Figures 4.3 and 4.8 respectively), are applied to TSP and QAP, and are proposed based on these contributions. The improvements are confirmed computationally and statistically.

In the second contribution, a more robust indication is achieved using the machine learning based indicator denoted as *ACO*ustic (see Section 5.2). The idea of inventing this indicator is mitigated from the acoustic mimicry phenomena in natural ants-parasites systems. Using *ACO*ustic, the exploration process is redefined using the number of clusters as a metric. The amount of exploration is measured by the degree of relatedness between artificial ants. In spite of the computational cost of the application of *ACO*ustic, it has shown empirically to be a more robust indicator. The significance of this proposal is not only in the improvement in the robustness of the indication, but also in the attempt of modeling the sounds of ants. The development of *ACO*ustic can be invested in different ways and lead to propose more advanced E&E components not for ACO algorithms only, but for other metaheuristics.

The third contribution concerns the proposal of three rewards assignment strategies for adaptive parameters' selection. There are the quality-based, the exploration-based and the unified strategies (see Subsections 5.7.1, 5.7.2 and 5.7.3 respectively). After a proper value for a particular parameter is selected proportionally, it is applied to the optimization process. The impact of the application is transformed into numerical rewards. One of the proposed strategies can be applied to calculate the rewards based on the reported impact. The experimental results showed that the quality-based reward assignment strategy has the more impact than others.

The three contributions are merged, in the unified RACO (see Chapter Six), and are empirically assessed to ensure that the quality of solutions produced does not worsen. The results showed that the contributed components operate more

harmoniously. The final algorithmic approach has been compared with eight algorithms in the application of TSP and six algorithms in the application of QAP.

7.2 Future Work

Automating the exploration and exploitation is promoted by the RACO proposal. RACO is assembled from several methods which are combined with the standard version of MMAS. Several more efficient and faster converging ACO variants exist, such as the PACO algorithm. Recent empirical studies confirmed that the pheromone evaporation in MMAS took a long time because of the need to divide every arc in the pheromone matrix in every iteration. This situation does not exist in PACO. Emerging RACO with the PACO approach in the near future can achieve better results than when combined with the standard algorithm.

Coming back to the individual methods combined with RACO, some of the methods are designed in a very independent way, and it can be considered that the application to any of the ACO variants is just as a baseline. For the RLS proposal, it can be applied for any local search algorithm whether it is a stand-alone or hybridized with another algorithm. If this idea were to be implemented, there are several considerations need to be taken such as the criteria under which the solutions will be added/deleted to/from the population vector.

For the *ACO*ustic proposal in reporting the performance of ACO algorithms, it is important to examine more fitness landscapes for other CO problems such as vehicle routing problem (VRP) and car sequencing problem (CSP). It is suspected that new

insights can be discovered from analyzing such landscapes using the relatedness concept to indicate the amount of exploration. On the other hand, *ACO*ustic can be used for controlling the dynamic transformation between exploration and exploitation for other metaheuristics such as the Artificial Bee Colony (ABC) algorithm. For ABC, *ACO*ustic can be utilized as a machine learning trigger for controlling the dynamic for the transformation between the onlookers and the scouts, or can be used to determine the optimal point to immigrate the current neighborhood structure.

Another path for further work is the application of the APS_{ACO} strategy for other ACO variants such as ACS or other swarm intelligence algorithms. The Particle Swarm Optimization (PSO) algorithm is a good candidate due to the similar number of parameters used for adjusting the exploration and exploitation within PSO. Along the same line, there are other strategies for deriving the numerical rewards in relation with the size of population memory referring to the pheromone management in the PACO approach.

A major drawback of the final recommended RACO method is that its application remains limited. It will be very useful to apply general enough adaptive parameter selection methods such as the ones applied in evolutionary algorithms. With the aid of *ACO*ustic, the fitness improvement used in assessing the impact of the parameters' application can be put in relation to the diversity of population in order to efficiently tackle multimodal problems especially when PSO is utilized as an underline algorithm or the adaptive operator selection (AOS) paradigm in

evolutionary algorithms, when it is used as a meta-optimization algorithm for the parameters' selection problem, particularly, in RACO and, generally, in ACO. In order to generalize the scientific contributions proposed in this thesis, there is intention to prepare freely available source codes with well-designed interface of the proposed approach. More and more combinatorial optimization problems, e.g. VRP and CSP, can be modeled based on the construction graph concept.

REFERENCES

Aleti, A. (2012). *An adaptive approach to controlling parameters of evolutionary algorithms*. Unpublished doctoral dissertation. Swinburne University of Technology, Melbourne Australia.

Adenso-Diiaz, B., & Laguna, M. (2006). Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, *54*(1), 99–114.

Afshar, M. (2005). A new transition rule for ant colony optimization algorithms: Application to pipe network optimization problems. *Engineering Optimization*, *37*(5), 525–540.

Alaya, I., Solnon, C., & Ghedira, K. (2004). Ant algorithm for the multi-dimensional knapsack problem. In B. Flipic & J. Silic (Eds.), *Proceedings of BIOMA'2004: The International Conference on Bioinspired Optimization Methods and Their Applications* (pp. 63–72). Ljubljana, Slovenia: Jozef Stefan Institute.

Aljanaby, A., Ku-Mahamud, K. R., & Norwawi, N. M. (2010). Interacted multiple ant colonies optimization framework: An experimental study of the evaluation and the exploration techniques to control the search stagnation. *International Journal of Advancements in Computing Technology*, *2*(1), 78–85.

Alobaedy, M. M., & Ku-Mahamud, K. R. (2015). Strategic Oscillation for Exploitation and Exploration of ACS Algorithm for Job Scheduling in Static Grid Computing. In *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)* (pp. 87–92). Johor: IEEE.

Altiparmak, F., & Karaoglan, I. (2007). A genetic ant colony optimization approach for concave cost transportation problems. In *Proceedings of CEC 2007: The IEEE Congress on Evolutionary Computation* (pp. 1685–1692). Stamford, Singapore: IEEE.

Amir, C., Badr, A., & Farag, I. (2007). A fuzzy logic controller for ant algorithms. *Computing and Information Systems*, *11*(2), 26–34.

Anghinolfi, D., Boccalatte, A., Paolucci, M., & Vecchiola, C. (2008). Performance evaluation of an adaptive ant colony optimization applied to single machine scheduling. In L. Xiaodong, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, … S. Yuhui (Eds.), *Lecture Notes in Computer Science: Simulated Evolution and Learning* (Vol. 5361, pp. 411–420). Heidelberg, Germany: Springer.

Angus, D. J. (2008). *Niching ant colony optimisation*. Unpublished doctoral dissertation.Swinburne University of Technology Melbourne, Australia.

Baghel, M., Agrawal, S., & Silakari, S. (2012). Survey of metaheuristic algorithms for combinatorial optimization. *International Journal of Computer Applications*, *58*(19), 21–31.

Balaprakash, P., Birattari, M., & Stützle, T. (2007). Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In T. Bartz-Beielstein, M. J. B. Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, & M. Sampels (Eds.), *Lecture Notes in Computer Science: Hybrid Metaheuristics* (Vol. 4771, pp. 108–122). Heidelberg, Germany: Springer.

Barbero, F., Patricelli, D., Witek, M., Balletto, E., Casacci, L. P., Sala, M., & Bonelli, S. (2012). Myrmica ants and their butterfly parasites with special focus on the acoustic communication. *Psyche: A Journal of Entomology*.

Barbero, F., Thomas, J., Bonelli, S., Balletto, E., & Schönrogge, K. (2009). Queen ants make distinctive sounds that are mimicked by a butterfly social parasite. *Science*, *323*(5915), 782–785.

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. ., & Stewart, W. R. (1995). Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, *1*, 9–32.

Battiti, R., & Birattari, M. (2013). *The LION way. Machine Learning plus Intelligent Optimization*. Los Angeles: Lionsolver Inc.

Battiti, R., Brunato, M., & Mascia, F. (2008). *Reactive search and intelligent optimization*. U.S.A.: Springer.

Battiti, R., & Protasi, M. (2001). Reactive local search for the maximum clique problem. In R. Battiti & M. Protasi (Eds.), *Algorithmica* (Vol. 29, pp. 610–637). Heidelberg, Germany: Springer.

Bertsimas, D., Brown, D. B. D., & Caramanis, C. (2011). Theory and Applications of Robust Optimization. *SIAM Review*, *53*(3), 464–501.

Beer, C., Hendtlass, T., & Montgomery, J. (2012). Improving exploration in ant colony optimization with antennation. In *2012 IEEE Congress on Evolutionary Computation*. Brisbane, Australia. Retrieved from http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6252923

Bin, Y., Zhongzhen, Y., & Baozhen, Y. (2009). An improved ant colony optimization for vehicle routing problem. *European Journal of Operational Research*, *196*(1), 171–176.

Birattari, M., Stützle, T., Paquete, L., & Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In W. B. Langdon, E. Cantü-Paz, K. E. Mathias, R. Roy, D. Davis, R. Poli, … N. Jonoska (Eds.), *Proceedings of GECCO'02:*

The Genetic and Evolutionary Computation Conference (pp. 11–18). San Francisco, CA, USA: Morgan Kaufmann.

Blum, C. (2002). ACO applied to group shop scheduling: A case study on intensification and diversification. In M. Dorigo, G. Di-Caro, & M. Sampels (Eds.), *Lecture Notes in Computer Science (LNCS): Ant algorithms* (Vol. 2463, pp. 14–27). Heidelberg, Germany: Springer.

Blum, C. (2005a). Ant colony optimization: Introduction and recent trends. *Physics Of Life Reviews*, *2*(4), 353–373.

Blum, C. (2005b). Beam-ACO—hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, *32*(6), 1565–1591.

Blum, C., & Blesa, M. J. (2005). New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Computers & Operations Research*, *32*(6), 1355–1377.

Blum, C., & Dorigo, M. (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, *34*(2), 1161–1172.

Blum, C., Puchinger, J., Raidl, G. R., & Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, *11*, 4135–4151.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, *35*(3), 268–308.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. New York, USA: Oxford Univ. Press.

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, *237*(2013), 82–117.

Bui, T. N., & Zrncic, C. M. (2006). An ant-based algorithm for finding degree-constrained minimum spanning tree. In M. Keijzer, J. A. Foster, D. V. Arnold, A. Hernández-Aguirre, V. Babovic, G. S. Hornby, … D. Thierens (Eds.), *Proceedings of GECCO'06: The 8th Annual Conference on Genetic And Evolutionary Computation* (Vol. 1, pp. 11–18). New York, USA: ACM Press.

Bullnheimer, B., Hartl, R. F., & Straub, C. (1997). *A new rank based version of the ant system - A computational study* (*working paper no.*1 ed.). Vienna: Institute of Management Science, University of Vienna.

Burkard, R. ., Cela, E., Karisch, S. E., & Rendl, F. (1997). QAPLIB - A quadratic assignment problem library. *Journal of Global Optimization*, (10), 391–403. Retrieved from http://www.opt.math.tu-graz.ac.at/qaplib/.

Carterette, B. (2011). An analysis of NP-completeness in novelty and diversity ranking. *Information Retrieval*, *14*(1), 89–106. doi:10.1007/s10791-010-9157-1.

Chen, W., Bian, W., & Zeng, R. (2013). Research on convergence of ACO subset algorithms. *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, *32*(2), 649–660.

Chusanapiputt, S., Nualhong, D., Jantarang, S., & Phoomvuthisarn, S. (2006). Selective self-adaptive approach to ant system for solving unit commitment problem. In M. Keijzer, J. A. Foster, D. V. Arnold, A. Hernández-Aguirre, V. Babovic, G. S. Hornby, … D. Thierens (Eds.), *Proceedings of GECCO '06: The 8th Annual Conference on Genetic and Evolutionary Computation* (Vol. 2, pp. 1729–1736). New York, USA: ACM Press.

Colas, S., & Monmarch, N. (2008). Artificial ants for the optimization of virtual keyboard arrangement for disabled people. In N. Monmarché, E.-G. Talbi, P. Collet, M. Schoenauer, & E. Lutton (Eds.), *Lecture Notes in Computer Science (LNCS): Artificial Evolution* (Vol. 4926, pp. 87–99). Heidelberg, Germany: Springer.

Collings, J., & Kim, E. (2014). A distributed and decentralized approach for ant colony optimization with fuzzy parameter adaptation in traveling salesman problem. In *2014 IEEE Symposium on Swarm Intelligence (SIS)* (pp. 1– 9). Florida, U.S.A. doi:10.1109/SIS.2014.7011805

Colorni, A., Dorigo, M., Maniezzo, V., Elettronica, D., & Milano, P. (1991). Distributed optimization by ant colonies. In *Proceedings of ECAL91: The European Conference on Artificial Life* (pp. 134–142). Paris, France: Elsevier Publishing.

Cordon, O., Herrera, F., & Stützle, T. (2002). A review on the ant colony optimization metaheuristic: Basis, models and new Trends. *Mathware and Soft Computing*, *9*(2-3), 141–175.

Cordon, O., Herrera, I. F. de V. F., & Moreno, L. (2000). A new ACO model integrating evolutionary computation concepts: The best-worst ant system. In M. Dorigo, M. Middendorf, & T. Stützle (Eds.), *Preceedings of ANTS'2000: From Ant Colonies to Artificial Ants: A Series of International Workshops On Ant Algorithims* (pp. 22–29). Brussels, Belgium: IRIDIA, Universite Libre de Bruxelles.

Çunkaş, M., & Özsağlam, M. Y. (2009). A Comparative Study on Particle Swarm Optimization and Genetic Algorithms for Traveling Salesman Problems. *Cybernetics And Systems*, *40*(6), 490–507. doi:10.1080/01969720903068435.

Deneubourg, J.-L., Aron, S., Goss, S., & Pasteels, J. M. (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, *3*(2), 159–168.

Deren, L., Kaichang, D., & Deyi, L. (2000). Knowledge representation and uncertainty reasoning in GIS based on cloud models. In *The 9th International Symposium On Spatial Data Handling*. Beijing, China. Retrieved from http://shoreline.eng.ohio- state.edu /dkc/SDH2000_Li_Di.pdf

Di-Caro, G., & Dorigo, M. (1998). AntNet: Distributed stigmergetic control for communications networks. *JAIR: Journal of Artificial Intelligence Research*, *9*(1), 317–365.

Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Unpublished doctoral dissertation. Politecnico di Milano, Italy.

Dorigo, M. (2007). Ant colony optimization. *Scholarpedia*, *2*(3), 1–12.

Dorigo, M., & Blum, C. (2005). Ant colony optimization theory: A survey. *Theoretical Computer Science*, *344*(2-3), 243–278.

Dorigo, M., & Di-Caro, G.. (1999). The ant colony optimization meta-heuristic. In C. David, M. Dorigo, & F. Glover (Eds.), *New Ideas in Optimization*. London: McGraw-Hill.

Dorigo, M., Di-Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life*, *5*(2), 137–172.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system : A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, *1*(1), 53–66.

Dorigo, M., Gambardella, L. M., Middendorf, M., & Stützle, T. (2002). Special section on ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *6*(4), 317–319.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy* (Tech. Rept. 91-016). Milano, Italy: Laboratorio di Calcolatori, Dipartimento di Elettronica, Politecnico di Milano.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems*, *26*(1), 29–41.

Dorigo, M., & Socha, K. (2007). Ant colony optimization. In T. F. Gonzalez (Ed.), *Handbook of approximation algorithms and metaheuristics* (pp. 1–25). Boca Raton, Florida: Chapman & Hall/CRC.

Dorigo, M., & Stützle, T. (2004). *Ant colony optimization*. Cambridge, MA, USA: MIT Press.

Dorigo, M., & Stützle, T. (2010). Ant colony optimization: Overview and recent advances. In M. Gendreau & J. Potvin (Eds.), *Handbook of Metaheuristics* (Vol. 146, pp. 227–263). New York, USA: Springer US.

Eiben, A., Michalewicz, Z., Schoenauer, M., & Smith, J. (2007). Parameter control in evolutionary algorithms. In F. G. Lobo, C. F. Lima, & Z. Michalewicz (Eds.), *Studies In Computational Intelligence (SCI): Parameter Setting in Evolutionary Algorithms* (Vol. 54, pp. 19–46). Heidelberg, Germany: Springer.

Eiben, A., & Jelasity, M. (2002). A critical note on experimental research methodology in EC. In *Proceedings of CEC'02: The 2002 Congress on Evolutionary Computation* (Vol. 2, pp. 582–587). Washington, DC, USA: IEEE Computer Society.

Eiben, A., & Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, *1*(1), 19–31. doi:10.1016/j.swevo.2011.02.001.

Eswaramurthy, V. P., & Tamilarasi, A. (2009). Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *40*(9-10), 1004–1015.

Failho, A. R. S. (2010). *Adaptive Operator Selection for Optimization*. Unpublished doctoral dissertation. Ecole Doctorale d' Informatique, Universite Paris, Paris, France.

Favaretto, D., Moretti, E., & Pellegrini, P. (2009). On the explorative behavior of MAX–MIN ant system. In T. Stützle, M. Birattari, & H. H. Hoos (Eds.), *Lecture Notes in Computer Science (LNCS): Engineering Stochastic Local Search Algorithms* (Vol. 5752,, pp. 115–119). Heidelberg, Germany: Springer.

Fletcher, R. (1997). *Practical methods of optimization*. Chichester, England: Weley & Sons Ltd.

Forster, M., Bickel, B., Hardung, B., & Gabriella, K. (2007). Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In D. Thierens, H.-G. Beyer, J. Bongard, J. Branke, J. A. Clark, D. Cliff, … I. Wegener (Eds.), *Proceedings of GECCO 2007: The 9th Annual Conference on*

Genetic And Evolutionary Computation (pp. 1991–1998). London, UK: ACM Press.

Gaertner, D., & Clark, K. (2005). On optimal parameters for ant colony optimization algorithms TSP classifications. In H. Arabnia & R. Joshua (Eds.), *Proceedings of IC-AI 2005: The International Conference on Artificial Intelligence* (Vol. 2, pp. 83–89). Las Vegas, Nevada, USA: CSREA Press.

Gambardella, L. M., & Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In A. Prieditis & S. J. Russell (Eds.), *The Morgan Kaufmann Series in Machine Learning: Proceedings of The 12th International Conference on Machine Learning* (pp. 252–260). California: Morgan Kaufmann.

Gambardella, L. M., Montemanni, R., & Weyland, D. (2012). Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, *220*(3), 831–843. doi:10.1016/j.ejor.2012.02.038

Gambardella, L. M., Taillard, E. D., & Dorigo, M. (1999). Ant Colonies for the Quadratic Assignment Problem. *The Journal of The Operational Research Society*, *50*(2), 167. doi:10.2307/3010565

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA, USA: Freeman.

Garnier, S., Gautrais, J., & Theraulaz, G. (2007). The biological principles of swarm intelligence. *Swarm Intelligence*, *1*(1), 3–31.

Garro, B. A., Sossa, H., Vázquez, R. A., Juan, A., Batiz, D. D., & Othon, M. De. (2007). Evolving ant colony system for optimizing path planning in mobile robots. In *Proceedings of CERMA 2007: The 4th Congress of Electronics, Robotics and Automotive Mechanics* (pp. 444–449). Morelos, México: IEEE Computer Society Press.

Gendreau, M., & Potvin, J.-Y. (2010). Handbook in metaheuristics. New York, USA: Springer.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*(2), 95–99.

Guntsch, M. (2004). *Ant algorithms in stochastic and multi-criteria environments*. Unpublished doctoral dissertation. Universität Fridericiana zu Karlsruhe, Karlsruhe, Germany.

Guntsch, M., & Middendorf, M. (2002). A population based approah for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, & G. R. Raidl (Eds.), *Lecture*

Notes in Computer Science: Applications of Evolutionary Computing (Vol. 2279, pp. 72–81). Heidelberg, Germany: Springer.

Hooker, J. N. (1995). Testing heuristics: We have it all wrong. *Journal of heuristics*, *1*(1), 33–42.

Hoos, H. H., & Stützle, T (2005). *Stochastic local search: Foundations and applications*. U.S.A.: Elsevier Inc.

Hutter, F., & Leyton-brown, K. (2009). ParamILS : An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, *36*, 267–306.

Johnson, D. S. (2001). A Theoreticians guide for experimental analysis of algorithms. In M. H. Goldwasser & C. C. McGeoch (Eds.), *Data Structures, Near Neighbor Searches, and Methodology: Proceedings of The 5th and 6th DIMACS Implementation Challenges* (Vol. 59, pp. 215–250). U.S.A.: American Mathematical Society.

Johnson, D. S., & McGeoch, L. A. (2007). Experimental Analysis of Heuristics for the ATSP. In G. Gutin & A. P. Punnen (Eds.), *The Traveling Salesman Problem And Its Variations* (pp. 445–487). Springer. doi:10.1007/b101971.

Kawamura, H., Yamamoto, M., & Suzuki, K. (2000). Multiple ant colonies algorithm based on colony level. *IEICE TRANSACTIONS On Fundamentals of Electronics, Communications and Computer Sciences*, *E83*(2), 371–379.

Khichane, M., Albert, P., & Solnon, C. (2009). An ACO-based reactive framework for ant colony optimization : First experiments on constraint satisfaction problems. In T. Stützle (Ed.), *Lecture Notes in Computer Science (LNCS): Learning and Intelligent Optimization* (Vol. 5851, pp. 119–133). Heidelberg, Germany: Springer.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*(4598), 671–80.

Kocer, H. E., & Akca, M. R. (2014). an Improved Artificial Bee Colony Algorithm With Local Search for Traveling Salesman Problem. *Cybernetics and Systems*, *45*(8), 635–649. doi:10.1080/01969722.2014.970396.

Korte, B., & Vygen, J. (2006). *Combinatorial optimization: Theory and algorithms*. Heidelberg, Germany: Springer.

Kov, O., & Skrbek, M. (2008). Ant Colony Optimization with Castes. In V. Kůrková, R. Neruda, & J. Koutník (Eds.), *Lecture Notes in Computer Science (LNCS): Artificial Neural Networks (ICANN 2008): Proceedings of the 18th*

International Conference (Vol. 5163, pp. 435–442). Heidelberg, Germany: Springer.

Ku-Mahamud, K. R., & Alobaedy, M. M. (2013). New heuristic function in ant colony system algorithm for optimization. In A. Kanarachos (Ed.), *Proceedings of MAMECTIS '13: The 15th International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems* (pp. 13–18). Lemesos, Cyprus: WSEAS.

Lafayette, W. (2001). Experimental evaluation of heuristic optimization algorithms: A tutorial, *304*, 261–304.

Lawler, E. L. (1963). The quadratic assignment problem. Management Science, 9(4), 586–599

Lawler, E. L., Lenstra, J. K., Kan, A. H. G., & Shmoys, D. B. (1985). The travelling salesman problem. New Jersey, U.S.A.: John Wiley & Sons.

Lin, Y., & Middendorf, M. (2013). Simple probabilistic population based optimization for combinatorial optimization. In *SIS: IEEE Symposium on Swarm Intelligence* (pp. 213–220). Singapore: IEEE. doi:10.1109/SIS.2013.6615181

Liu, X., & Yang, C. (2011). Optimization of Vehicle Routing Problem Based on Max-Min Ant System with Parameter Adaptation. In *2011 Seventh International Conference on Computational Intelligence And Security*. China, Hainan: IEEE.

Liu, Y., Ang, G., Chen, H., Zhao, Z., Zhu, X., & Liu, Z. (2011). An adaptive fuzzy ant colony optimization for feature selection. *Journal Of Computational Information Systems*, *4*(7), 1206–1213.

Lopez-Ibanez, M. (2010). Ant Colony Optimization. In M. Pelikan & J. Branke (Eds.), *Proceedings of GECCO'10: The 12th Annual Conference Companion On Genetic and Evolutionary Computation* (pp. 2353–2384). New York, NY, USA: ACM.

Lopez-Ibanez, M., & Stützle, T. (2011). The automatic design of multi-objective ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation*, *16*(6), 861 – 875. doi:10.1109/TEVC.2011.2182651.

Lopez-Ibanez, M., & Stützle, T. (2014). Automatically improving the anytime behaviour of optimisation algorithms. *European Journal of Operational Research*, *235*(3), 569– 582.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS: Journal on Computing*, *11*(4), 1–22.

Martens, D., Backer, M. De, Haesen, R., Vanthienen, J., & Snoeck, M. (2007). Classification with ant colony optimization. *IEEE Transactions on Evolutionary Computation*, *11*(5), 651–665.

Martens, D., Baesens, B., & Fawcett, T. (2011). Editorial survey : swarm intelligence for data mining. Machine Learning, 82(April 2010), 1–42. doi:10.1007/s10994-010-5216-5.

Maur, M., Stützle, T., & López-Ibáñez, M. (2010). Pre-scheduled and adaptive parameter variation in MAX-MIN Ant System. In *2010 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1 – 8). Spain, Barcelona: IEEE. doi:10.1109/CEC.2010.5586332.

Melo, L., Pereira, F., & Costa, E. (2010). MC-ANT: A multi-colony ant algorithm. In P. Collet, N. Monmarché, P. Legrand, M. Schoenauer, & E. Lutton (Eds.), *Lecture Notes in Computer Science: Artifical Evolution* (Vol. 5975, pp. 25–36). Heidelberg, Germany: Springer.

Merkle, D., Middendorf, M., & Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, *6*(4), 333–346.

Merkle, D., & Middendorf, M. (2001). Prospects for dynamic algorithm control: Lessons from the phase structure of ant scheduling algorithms. In *GECCO 2001-Genetic and Evolutionary Computation Conference Workshop Program*. San Francisco, California, USA.

Merkle, D., & Middendorf, M. (2005). Swarm intelligence. In E. K. Burke & G. Kendall (Eds.), *Search Methodologies* (pp. 401–435). U.S.A.: Springer.

Meyer, B. (2004). Convergence control in ACO. In M. Keijzer (Ed.), *Proceedings of GECCO'04: Genetic and Evolutionary Computation Conference, seattle, washington, late-breaking paper available on CD*. X-CD Technologies. Retrieved from http://www.cs.bham.ac.uk/~wbl/biblio/gecco2004/LBP035.pdf

Middendorf, M., Reischle, F., & Schmeck, H. (2000). Information exchange in multi colony ant algorithms. In J. Rolim (Ed.), *Lecture Notes in Computer Science (LNCS): Parallel and Distributed Processing* (Vol. 1800, pp. 645–652). Heidelberg, Germany: Springer.

Mohan, B. C., & Baskaran, R. (2012). A survey : Ant Colony Optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, *39*(4), 4618–4627. doi:10.1016/j.eswa.2011.09.076.

Monteiro, M., Fontes, D., & Fontes, F. (2012). Ant colony optimization: A literature survey (working paper, no. 474). Retrieved from FEP- Faculty of Engineering, University of Porto: http://www.fep.up.pt/investigacao/workingpapers/wp474.pdf

Moret, B. M. E. (2001). Algorithms and experiments: The new (and old) methodology, *7*(5), 434–446.

Neyoy, H., Castillo, O., & Soria, J. (2013). Dynamic fuzzy logic parameter tuning for ACO and its application in TSP problems. In O. Castillo, P. Melin, & J. Kacprzyk (Eds.), *Studies in Computational Intelligence (SCI): Recent Advances On Hybrid Intelligent Systems* (Vol. 451, pp. 259–271). Heidelberg, Germany: Springer.

Neyoy, H., Castillo, O., & Soria, J. (2015). *Fuzzy logic for dynamic parameter tuning in ACO and its application in optimal fuzzy logic controller design*. (O. Castillo & P. Melin, Eds.). Springer. doi:10.1007/978-3-319-10960-2_1

Olivas, F., Valdez, F., & Castillo, O. (2015). Dynamic parameter adaptation in Ant Colony Optimization using a fuzzy system for TSP problems. In *16th World Congress of The International Fuzzy Systems Association (IFSA) and The 9th Conference of The European Society for Fuzzy Logic and Technology (EUSFLAT) Dynamic* (pp. 765–770). Gijón, Asturias, Spain: Published by Atlantis Press.

Oliveira, S., Stützle, T., Roli, A., & Dorigo, M. (2011). A Detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In *Proceedings of GECCO'11: Genetic and Evolutionary Computation Conference* (pp. 13–14). doi:10.1145/2001858.2001866

Osman, I., & Laporte, G. (1996). Metaheuristics: A bibliography. In I. Osman & G. Laporte (Eds.), *Annals of Operational Research* (Vol. 63, pp. 513–628). Baarn/Kluwer, Netherlands: Baltzer Science Publishers.

Pellegrini, P. (2006). *ACO : parameters , exploration and quality of solutions*. Unpublished doctoral dissertation. Universit`a Ca' Foscari Venezia, Venezia, Italy.

Pellegrini, P., & Favaretto, D. (2012). Quantifying the exploration performed by metaheuristics. *Journal of Experimental & Theoretical Artificial Intelligence*, *24*(2), 247–266. doi:10.1080/0952813X.2012.656327

Pellegrini, P., Favaretto, D., & Moretti, E. (2009). Exploration in stochastic algorithms : An application on MAX-MIN ant system. In N. Krasnogor, M. B. Melián-Batista, J. A. M. Pérez, J. M. Moreno-Vega, & D. A. Pelta (Eds.), *Studies in computational intelligence (SCI): Nature Inspired Cooperative*

Strategies for Optimization (NICSO 2008) (Vol. 236, pp. 1–13). Heidelberg, Germany: Springer.

Pellegrini, P., Stützle, T., & Birattari, M. (2010a). *Off-line vs . on-line tuning : A study on MAX – MIN ant system for the TSP* (TR/IRIDIA/2010-009). Bruxelles, Belgium: IRIDIA, Universite Libre de Bruxelles.

Pellegrini, P., Stützle, T., & Birattari, M. (2010b). *Tuning Max-Min ant system with off-line and on-line methods* (TR/IRIDIA/2010-009). Bruxelles, Belgium: IRIDIA, Universite Libre de Bruxelles.

Pellegrini, P., Stützle, T., & Birattari, M. (2012). A critical analysis of parameter adaptation in ant colony optimization. *Swarm Intelligence*, *6*(1), 23–48.

Perez-Caceres, L., Lopez-Ibanez, M., & Stützle, T. (2014). Ant colony optimization on a budget of 1000. In M. Dorigo, M. Birattari, S. Garnier, H. Hamann, M. M. de Oca, C. Solnon, & T. Stützle (Eds.), *Lecture Notes in Computer Science (LNCS): Swarm Intelligence* (Vol. 8667, pp. 50–61). Springer Berlin Heidelberg. doi:10.1007/978-3-319-09952-1_5

Pilat, M. L., & White, T. (2002). Using genetic algorithms to optimize ACS-TSP. In M. Dorigo, G. Di-Caro, & M. Sampels (Eds.), *Lecture Notes in Computer Science: Ant Algorithms* (Vol. 2463, pp. 282–287). Heidelberg, Germany: Springer.

Rahmani, P., Dadbakhsh, M., & Gheisari, S. (2012). Improved MACO approach for grid scheduling. In *Proceedings of ICIII 2012: The International Conference on Industrial and Intelligent Information* (Vol. 31, pp. 135–142). Singapore: IACSIT Press.

Rajaraman, A., & Ullman, J. D. (2012). *Mining of massive datasets*. U.S.A.: Cambridge University Press.

Rappos, E., & Hadjiconstantinou, E. (2004). An ant colony heuristic for the design of two-edge connected flow networks. In M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, & T. Stützle (Eds.), *Lecture Notes in Computer Science* (Vol. 3172, pp. 270–277). Heidelberg, Germany: Springer.

Reinelt, G. (1991). TSPLIB-A traveling salesman problem library. *ORSA Journal On Computing*. Retrieved from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Rothlauf, F. (2011). *Design of modern heuristics: Principles and application*. Mainz, Germany: Springer.

Russell, S. J., & Norvig, P. (2010). Artificial intelligence: A modern approach. New Jersey, U.S.A.: Prentice Hall.

Seo, M. (2009). *Applications of the ant colony optimization algorithm in combinatorial optimization*. Unpublished doctoral dissertation. Pennsylvania State University, Pennsylvania State, U.S.

Shyong, J. S., Pengyeng, Y., & Bertrand, M. T. L. (2004). An ant colony optimization algorithm for the minimum weight vertex cover problem. In J. S. Shyong, Y. Pengyyeng, & M. T. L. Bertrand (Eds.), *Annals of Operations Research* (Vol. 131, pp. 283–304). U.S.A.: Kluwer Academic Publishers.

Solimanpur, M., Vrat, P., & Shankar, R. (2005). An ant algorithm for the single row layout problem in flexible manufacturing systems. *Computers & Operations Research*, *32*(3), 583–598.

Solnon, C. (2010). *Ant colony optimization and constraint programming*. U.S.A.: Weley & Sons Ltd. doi:10.1002/9781118557563

Solnon, C., & Fenet, S. (2005). A study of ACO capabilities for solving the maximum clique problem. *Journal of Heuristics*, *12*(3), 155–180.

Stützle, T. (1999). *Local search algorithms for combinatorial problems: Analysis , improvements , and new applications*. Unpublished doctoral dissertation. Technische Universit, Darmstadt, German.

Stützle, T. (2004). ACOTSP, Version 1.03. Retrieved from http://www.aco-metaheuristic.org/aco-code

Stützle, T., & Hoos, H. (1999). MAX-MIN ant system and local search for combinatorial optimization problems. In S. Vob, S. Martello, I. H. Osman, & C. Roucairol (Eds.), *Meta-Heuristics* (pp. 313–329). U.S.A.: Springer.

Stützle, T., & Hoos, H. H. (1998). Improvements on ant-system: Introducing MAX-MIN ant system. In T. Stützle & H. H. Hoos (Eds.), *Artificial Neural Nets and Genetic Algorithms* (pp. 245–249). Vienna: Springe.

Stützle, T., & Hoos, H. H. (2000). MAX-MIN ant system. *FGCS: Future Generation Computer Systems*, *16*(8), 889–914.

Stützle, T., & López-Ibáñez, M. (2013). Automatic (offline) algorithm configuration. In *GECCO'13: The Fifteenth Annual Conference Companion on Genetic and Evolutionary Computation Conference Companion* (pp. 893–918). New York, USA: ACM.

Stützle, T., López-Ibáñez, M., Pellegrini, P., Maur, M., Oca, M. M. De, Birattari, M., & Dorigo, M. (2012). Parameter adaptation in ant colony optimization. In Y. Hamadi, E. Monfroy, & F. Saubion (Eds.), *Autonomous Search* (pp. 191–215). Heidelberg, Germany: Springer.

Taillard, E. (2010). FANT, Retrieved from http://mistic.heig-vd.ch/taillard/codes.dir/fant_qap.c.

Talbi, E.-G. (2009). *Metaheuristics from design to implementation*. U.S.A.: Wiley.

Talbi, E.-G., Roux, O., Fonlupt, C., & Robillard, D. (2001). Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, *17*(4), 441–449.

Theraulaz, G., & Bonabeau, E. (1999). A brief history of stigmergy. *Artificial life*, *5*(2), 97–116.

Thomas, J. A., Schonrogge, K., Bonelli, S., Barbero, F., & Balletto, E. (2010). Corruption of ant acoustical signals by mimetic social parasites. *Communicative and Integrative Biology*, *3*(2), 169–171.

Venables, H., & Moscardini, A. (2006). An adaptive search heuristic for the capacitated fixed charge location problem. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli, & T. Stützle (Eds.), *Lecture Notes in Computer Science: Ant Colony Optimization and Swarm Intelligence* (Vol. 4150, pp. 348–355). Heidelberg, Germany: Springer.

Wang, Y. (2013). Adaptive Ant Colony Algorithm for the VRP Solution of Logistics Distribution. *Research Journal of Applied Sciences, Engineering and Technology*, *6*(5), 807–811.

Weixin, L., & Huanping, L. (2007). An adaptive parameter control strategy for ant colony optimization. In *CIS '07: The International Conference on Computational Intelligence and Security* (pp. 142–146). Harbin: IEEE.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80-83.

Yancang, L., & Wanqing, L. (2007). Adaptive ant colony optimization algorithm based on information entropy. *Fundamenta Informaticae*, *77*(3), 229–242.

Ziqiang, L., & Yi, Y. (2014). A hybrid artificial fish-school optimization algorithm for solving the quadratic assignment problem. In *2014 10th International Conference on Natural Computation (ICNC)* (pp. 1099–1104). China, Xiamen: IEEE. doi:10.1109/ICNC.2014.6975994.

Zhaoquan, C., Han, H., Yong, Q., & Xianheng, M. (2009). Ant colony optimization based on adaptive volatility rate of pheromone trail. *International Journal Of Communications, Network And System Sciences*, *02*, 792–796.

Zhifeng, H., Han, H., Yong, Q., & Ruichu, C. (2007). An ACO algorithm with adaptive volatility rate of pheromone trail. In Y. Shi, G. Albada, J. Dongarra, &

P. Sloot (Eds.), *Lecture Notes in Computer Science: Proceedngs of ICCS2007-The 7th International Conference in Computational Science* (Vol. 4490, pp. 1167–1170). Heidelberg, Germany: Springer.

Zhifeng, H., Ruichu, C., & Han, H. (2006). An adaptive parameter control strategy for aco. In *Proceedings of ICML 2006: The Fifth International Conference on Machine Learning and Cybernetics* (pp. 203–206). China, Dalian: IEEE Press.

Zhiyong, L., Yong, W., Jianping, Y., Youjia, Z., & Xu, L. (2008). A novel cloud-based fuzzy self-adaptive ant colony system. In *ICNC 2008: The 4th International Conference on Natural Computation* (pp. 460–465). Jinan, China: IEEE Computer Society.

Zhongzhen, Y., Bin, Y., & Chuntian, C. (2007). A parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering*, *22*(1), 44–55.

Zufferey, N. (2012). Metaheuristics: Some Principles for an Efficient Design. *Computer Technology and Application*, *3*(2012), 446–462.

Appendix A

Technical Details of the TSPLIB Files

A.1: TSPLIB File Format

1- TSPLIB File Specification Section:

* NAME : <string> Identifies the data file.
* TYPE : <string> Specifies the type of data. Possible types are
    * TSP Data for a symmetric traveling salesman problem
    * ATSP Data for an asymmetric traveling salesman problem
    * HCP Hamiltonian cycle problem data.
    * HPP Hamiltonian path problem data (not available in TSPLIB)
* COMMENT : <string> Additional comments (usually the name of the contributor or the creator of the problem instance is given here).
* DIMENSION : < integer> the number of nodes.
* EDGE_WEIGHT_TYPE : <string> Specifies how the edge weights (or distances) are given. The values are:
    * ATT Special distance function for problem att48 and att532
    * CEIL_2D Weights are Euclidean distances in 2-D rounded up
    * CEIL_3D Weights are Euclidean distances in 3-D rounded up
    * EUC_2D Weights are Euclidean distances in 2-D
    * EUC_3D Weights are Euclidean distances in 3-D
    * EXPLICIT Weights are listed explicitly in the corresponding section
    * GEO Weights are geographical distances in kilometres (TSPLIB). Coordinates are given in the form DDD.MM where DDD are the degrees and MM the minutes
    * GEOM Weights are geographical distances in meters (used for the world TSP). Coordinates are given in decimal form
    * GEO_MEEUS Weights are geographical distances in kilometres, computed according to Meeus' formula. Coordinates are given in the form DDD.MM where DDD are the degrees and MM the minutes
    * GEOM_MEEUS Weights are geographical distances, computed according to Meeus' formula. Coordinates are given in decimal form
    * MAN_2D Weights are Manhattan distances in 2-D
    * MAN_3D Weights are Manhattan distances in 3-D
    * MAX_2D Weights are maximum distances in 2-D
    * MAX_3D Weights are maximum distances in 3-D
* EDGE-WEIGHT_FORMAT : <string> Describes the format of the edge weights if they are given explicitly. The values are
    * FULL_MATRIX Weights are given by a full matrix
    * UPPER_ROW Upper triangular matrix (row-wise without diagonal entries)
    * LOWER_ROW Lower triangular matrix (row-wise without diagonal entries)
    * UPPER_DIAG_ROW Upper triangular matrix (row-wise including diagonal entries)
    * LOWER_DIAG_ROW Lower triangular matrix (row-wise including diagonal entries)
    * UPPER_COL Upper triangular matrix (column-wise without diagonal entries)
    * LOWER_COL Lower triangular matrix (column-wise without diagonal entries)
    * UPPER_DIAG_COL Upper triangular matrix (column-wise including diagonal entries)
    * LOWER_DIAG_COL Lower triangular matrix (column-wise including diagonal entries)
* EDGE_DATA_FORMAT : <string> Describes the format in which the edges of a graph are given, if the graph is not complete. The values are
    * EDGE_LIST The graph is given by an edge list
    * ADJ_LIST The graph is given by an adjacency list

* NODE_COORD_TYPE : <string> Specifies whether the coordinates are associated with each node (which, for example may be used for either graphical display or distance computations. The values are
* TWOD_COORDS Nodes are specified by coordinates in 2-D
* THREED_COORDS Nodes are specified by coordinates in 3-D
* NO_COORDS The nodes do not have associated coordinates
* DISPLAY_DATA_TYPE : <string> Specifies how a graphical display of the nodes can be obtained. The values are
* COORD_DISPLAY Display is generated from the node coordinates
* TWOD_DISPLAY Explicit coordinates in 2-D are given
* BO_DISPLAY No graphical display is possible.
The default value is COORD_DISPLAY if node coordinates are specifies and NO_DISPLAY otherwise. In the current implementation, however, the value has no significance.

2- TSPLIB File Data Section:
Depending on the choice of specifications some additional data may be required. These data are given corresponding data sections following the specification section. Each data section begins with the corresponding keyword. The length of the section is either explicitly known form the format specification, or the section is terminated by an appropriate end-of-section identifier.

* NODE_COORD_SECTION: Node coordinates are given in this section. Each line is of the form <integer> <real> <real> if NODE_COORD_TYPE is TWOD_COORDS or <integer> <real> <real> <real> if NODE_COORD_TYPE is THREED_COORDS. The integers give the number of the respective nodes. The real numbers are the associated coordinates.
* EDGE_DATA_SECTION: Edges of the graph are specified in either of the two formats allowed in the EDGE_DATA_FORAT entry. If a type is EDGE_LIST, then the edges are given as a sequence of lines of the form <integer> <integer> each entry giving the terminal nodes of some edge. The list is terminated by a -1. If the type is ADJ_LIST, the section consists of adjacency list for nodes. The adjacency list of a node x is specified as <integer> <integer> ... <integer> -1 where the first integer gives the number of node x and the following integers (terminated by -1) the numbers of the nodes adjacent to x. The list of adjacency lists are terminated by an additional -1.
* FIXED_EDGES_SECTION: In this section, edges are listed that are required to appear in each solution to the problem. The edges to be fixed are given in the form (per line) <integer> <integer> meaning that the edge (arc) from the first node to the second node has to be contained in a solution. This section is terminated by a -1.
* DISPLAY_DATA_SECTION:
* If DISPLAY_DATA_TYPE is TWOD_DISPLAY, the 2-dimensional coordinates from which a display can be generated are given in the form (per line) <integer> <real> <real> the integers specify the respective nodes and the real numbers give the associated coordinates. The contents of this section, however, have no significance in the current implementation.
* TOUR_SECTION: A tour is specified in this section. The tour is given by a list of integers giving the sequence in which the nodes are visited in the tour. The tour is terminated by a -1. Note: In contrast to the TSPLIB format, only one tour can be given in this section. The tour is used to limit the search (the last edge to be excluded in a non-gainful move must not belong to the tour). In addition, the Alpha field of its edges is set to -1.
* EDGE_WEIGHT_SECTION: The edge weights are given in the format specifies by the EDGE_WEIGHT_FORMAT entry. At present, all explicit data are integral and is given in one of the (self-explanatory) matrix formats, with explicitly known lengths.
* EOF Terminates input data. The entry is optional.

A.2: TSPLIB File Reading Implementation

```
void read_tsp(void)
/*
    FUNCTION: read TSP instance file
    INPUT:    instance name
    OUTPUT:   list of coordinates for all nodes
```

```
    COMMENTS: Instance files have to be in TSPLIB format, otherwise procedure fails
*/
{
   struct point {
              double x;
              double y;
   };
   FILE        *tsp_file;
   char        buf[LINE_BUF_LEN];
   long int    i, j;
   struct point *nodeptr;

   tsp_file = fopen("burm14.tsp", "r");
   if ( tsp_file == NULL ) {
          fprintf(stderr,"No instance file specified, abort\n");
          exit(1);
   }
   assert(tsp_file != NULL);
   printf("\nreading tsp-file %s ... \n\n", "burm14.tsp");

   fscanf(tsp_file,"%s", buf);
   while ( strcmp("NODE_COORD_SECTION", buf) != 0 ) {
          if ( strcmp("NAME", buf) == 0 ) {
             fscanf(tsp_file, "%s", buf);
             TRACE ( printf("%s ", buf); )
             fscanf(tsp_file, "%s", buf);
             strcpy(tsp_instance.name, buf);
             TRACE ( printf("%s \n", tsp_instance.name); )
             buf[0]=0;
          }
          else if ( strcmp("NAME:", buf) == 0 ) {
             fscanf(tsp_file, "%s", buf);
             strcpy(tsp_instance.name, buf);
             TRACE ( printf("%s \n", tsp_instance.name); )
             buf[0]=0;
          }
          else if ( strcmp("COMMENT", buf) == 0 ){
             fgets(buf, LINE_BUF_LEN, tsp_file);
             TRACE ( printf("%s", buf); )
             buf[0]=0;
          }
          else if ( strcmp("COMMENT:", buf) == 0 ){
             fgets(buf, LINE_BUF_LEN, tsp_file);
             TRACE ( printf("%s", buf); )
             buf[0]=0;
          }
          else if ( strcmp("TYPE", buf) == 0 ) {
             fscanf(tsp_file, "%s", buf);
             TRACE ( printf("%s ", buf); )
             fscanf(tsp_file, "%s", buf);
             TRACE ( printf("%s\n", buf); )
             if( strcmp("TSP", buf) != 0 ) {
                    fprintf(stderr,"\n Not a TSP instance in TSPLIB format !!\n");
                    exit(1);
             }
             buf[0]=0;
          }
```

```
else if ( strcmp("TYPE:", buf) == 0 ) {
   fscanf(tsp_file, "%s", buf);
   TRACE ( printf("%s\n", buf); )
   if( strcmp("TSP", buf) != 0 ) {
         fprintf(stderr,"\n Not a TSP instance in TSPLIB format !!\n");
         exit(1);
   }
   buf[0]=0;
}
else if( strcmp("DIMENSION", buf) == 0 ){
   fscanf(tsp_file, "%s", buf);
   TRACE ( printf("%s ", buf); );
   fscanf(tsp_file, "%ld", &n);
   tsp_instance.n = n;
   TRACE ( printf("%ld\n", n); );
   assert ( n > 2 && n < 6000);
   buf[0]=0;
}
else if ( strcmp("DIMENSION:", buf) == 0 ) {
   fscanf(tsp_file, "%ld", &n);
   tsp_instance.n = n;
   TRACE ( printf("%ld\n", n); );
   assert ( n > 2 && n < 6000);
   buf[0]=0;
}
else if( strcmp("DISPLAY_DATA_TYPE", buf) == 0 ){
   fgets(buf, LINE_BUF_LEN, tsp_file);
   TRACE ( printf("%s", buf); );
   buf[0]=0;
}
else if ( strcmp("DISPLAY_DATA_TYPE:", buf) == 0 ) {
   fgets(buf, LINE_BUF_LEN, tsp_file);
   TRACE ( printf("%s", buf); );
   buf[0]=0;
}
else if( strcmp("EDGE_WEIGHT_TYPE", buf) == 0 ){
   buf[0]=0;
   fscanf(tsp_file, "%s", buf);
   TRACE ( printf("%s ", buf); );
   buf[0]=0;
   fscanf(tsp_file, "%s", buf);
   TRACE ( printf("%s\n", buf); );
   if ( strcmp("EUC_2D", buf) == 0 ) {
         distance = round_distance;
   }
   else if ( strcmp("CEIL_2D", buf) == 0 ) {
         distance = ceil_distance;
   }
   else if ( strcmp("GEO", buf) == 0 ) {
         distance = geo_distance;
   }
   else if ( strcmp("ATT", buf) == 0 ) {
         distance = att_distance;
   }
   else
         fprintf(stderr,"EDGE_WEIGHT_TYPE %s not implemented\n",buf);
   strcpy(tsp_instance.edge_weight_type, buf);
```

```
            buf[0]=0;
        }
        else if( strcmp("EDGE_WEIGHT_TYPE:", buf) == 0 ){
            /* set pointer to appropriate distance function; has to be one of
               EUC_2D, CEIL_2D, GEO, or ATT. Everything else fails */
            buf[0]=0;
            fscanf(tsp_file, "%s", buf);
            TRACE ( printf("%s\n", buf); )
                printf("%s\n", buf);
            printf("%s\n", buf);
            if ( strcmp("EUC_2D", buf) == 0 ) {
                distance = round_distance;
            }
            else if ( strcmp("CEIL_2D", buf) == 0 ) {
                distance = ceil_distance;
            }
            else if ( strcmp("GEO", buf) == 0 ) {
                distance = geo_distance;
            }
            else if ( strcmp("ATT", buf) == 0 ) {
                distance = att_distance;
            }
            else {
                fprintf(stderr,"EDGE_WEIGHT_TYPE %s not implemented\n",buf);
                exit(1);
            }
            strcpy(tsp_instance.edge_weight_type, buf);
            buf[0]=0;
        }
        buf[0]=0;
        fscanf(tsp_file,"%s", buf);
    }
    if( strcmp("NODE_COORD_SECTION", buf) == 0 ){
        TRACE ( printf("found section containing the node coordinates\n"); )
            }
    else{
        fprintf(stderr,"\n\nSome error occurred finding start of coordinates from tsp file !!\n");
        exit(1);
    }
    if( (nodeptr = malloc(sizeof(struct point) * n)) == NULL )
        exit(EXIT_FAILURE);
    else {
        for ( i = 0 ; i < n ; i++ ) {
            fscanf(tsp_file,"%ld %lf %lf", &j, &nodeptr[i].x, &nodeptr[i].y );
        }
    }
    TRACE ( printf("number of cities is %ld\n",n); )
    TRACE ( printf("\n... done\n"); )
}
```

Appendix B

Statistical Details of the QAPLIB Files

B.1: Burkard QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Bur26a	26	5426670 (OPT)	(26 15 11 7 4 12 13 2 6 18 1 5 9 21 8 14 3 20 19 25 17 10 16 24 23 22)	
Bur26b	26	3817852	3753198	1.69 %
Bur26c	26	5426795	5361204	1.21 %
Bur26d	26	3821225	3758687	1.64 %
Bur26e	26	5386879	5334780	0.97 %
Bur26f	26	3782044	3733941	1.27 %
Bur26g	26	10117172	10055637	0.61 %
Bur26h	26	7098658	7045690	0.75 %

B.2: Christofides QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Chr12a	12	9552 (OPT)	(7,5,12,2,1,3,9,11,10,6,8,4)	
Chr12b	12	9742 (OPT)	(5,7,1,10,11,3,4,2,9,6,12,8)	
Chr12c	12	11156 (OPT)	(7,5,1,3,10,4,8,6,9,11,2,12)	
Chr15a	15	9896 (OPT)	(5,10,8,13,12,11,14,2,4,6,7,15,3,1,9)	
Chr15b	15	7990 (OPT)	(4,13,15,1,9,2,5,12,6,14,7,3,10,11,8)	
Chr15c	15	9504 (OPT)	(13,2,5,7,8,1,14,6,4,3,15,9,12,11,10)	
Chr18a	18	11098 (OPT)	(3,13,6,4,18,12,10,5,1,11,8,7,17,14,9,16,15,2)	
Chr18b	18	1534 (OPT)	(1,2,4,3,5,6,8,9,7,12,10,11,13,14,16,15,17,18)	
Chr20a	20	2192 (OPT)	(3,20,7,18,9,12,19,4,10,11,1,6,15,8,2,5,14,16,13,17)	
Chr20b	20	2298 (OPT)	(20,3,9,7,1,12,16,6,8,14,10,4,5,13,17,2,18,11,19,15)	
Chr20c	20	14142 (OPT)	(12,6,9,2,10,11,3,4,15,18,7,13,16,5,14,17,19,1,8,20)	
Chr22a	22	6156 (OPT)	(15,2,21,8,16,1,7,18,14,13,5,17,6,11,3,4,20,19,9,22,10,12)	
Chr22b	22	6194 (OPT)	(10,19,3,1,20,2,6,4,7,8,17,12,11,15,21,13,9,5,22,14,18,16)	
Chr25a	25	3796 (OPT)	(25,12,5,3,18,4,16,8,20,10,14,6,15,23,24,19,13,1,21,11,17,2,22,7,9)	

B.3: Elshafei QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Els19	19	17212548 (OPT)	(9,10,7,18,14,19,13,17,6,11,4,5,12,8,15,16,1,2,3)	

B.4: Eschermann QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Esc16a	16	68 (OPT)	(2,14,10,16,5,3,7,8,4,6,12,11,15,13,9,1)	
Esc16b	16	292 (OPT)	(6,3,7,5,13,1,15,2,4,11,9,14,10,12,8,16)	
Esc16c	16	160 (OPT)	(11,14,10,16,12,8,9,3,13,6,5,7,15,2,1,4)	
Esc16d	16	16 (OPT)	(14,2,12,5,6,16,8,10,3,9,13,7,11,15,4,1)	
Esc16e	16	28 (OPT)	(16,7,8,15,9,12,14,10,11,2,6,5,13,4,3,1)	
Esc16f	16	0 (OPT)	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)	
Esc16g	16	26 (OPT)	(8,11,9,12,15,16,14,10,7,6,2,5,13,4,3,1)	

Esc16h	16	996 (OPT)	(13,9,10,15,3,11,4,16,12,7,8,5,6,2,1,14)	
Esc16i	16	14 (OPT)	(13,9,11,3,7,5,6,2,1,15,4,14,12,10,8,16)	
Esc16j	16	8 (OPT)	(8,3,16,14,2,12,10,6,9,5,13,11,4,7,15,1)	
Esc32a	32	130	88	32.31 %
Esc32b	32	168	100	40.48 %
Esc32c	32	642	506	21.18 %
Esc32d	32	200	152	24.00 %
Esc32e	32	2 (OPT)	(1,2,5,6,8,16,13,19,9,32,7,22,24,20,4,12,3, 17,29,21,11,25,27,18,30,31,23,28,14,15,26,10)	
Esc32f	32	2 (OPT)	(1,2,5,6,8,16,10,7,9,28,30,4,32,31,22,12,3, 17,26,18,13,25,29,21,23,24,19,20,14,15,27,11)	
Esc32g	32	6 (OPT)	(14,15,16,12,11,26,30,10,25,8,29,22,31,28, 13,1,19,9,17,32,24,18,4,2,20,5,21,3,7,6,23,27)	
Esc32h	32	438	352	21.00 %
Esc64a	64	116	47	59.49 %
Esc128	128	64	2	96.86 %

B.5: Krarup QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Kra30a	30	88900		(OPT) (26,24,23,16,20,19,6,10,11,2,22,18,7,30,15,21,25,29,12,9,5,17,1,8, 13,28,14,3,4,27)
Kra30b	30	91420	(OPT)	(23,26,19,25,20,22,11,8,9,14,27,30,12,6,28, 24,21,18,1,7,10,29,13, 5,2,17,3,15,4,16)

B.6: Nugent QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Nug12	12	578 (OPT)	(12,7,9,3,4,8,11,1,5,6,10,2)	
Nug14	14	1014 (OPT)	(9,8,13,2,1,11,7,14,3,4,12,5,6,10)	
Nug15	15	1150 (OPT)	(1,2,13,8,9,4,3,14,7,11,10,15,6,5,12)	
Nug16a	16	1610	(OPT)	(9,14,2,15,16,3,10,12,8,11,6,5,7,1,4,13)
Nug16b	16	1240	(OPT)	(16,12,13,8,4,2,9,11,15,10,7,3,14,6,1,5)
Nug17	17	1732	(OPT)	(16,15,2,14,9,11,8,12,10,3,4,1,7,6,13,17,5)
Nug18	18	1930	(OPT)	(10,3,14,2,18,6,7,12,15,4,5,1,11,8,17,13,9,16)
Nug20	20	2570		(OPT) (18,14,10,3,9,4,2,12,11,16,19,15,20,8,13,17,5,7,1,6)
Nug21	21	2438		(OPT) (4,21,3,9,13,2,5,14,18,11,16,10,6,15,20,19,8,7,1,12,17)
Nug22	22	3596		(OPT) (2,21,9,10,7,3,1,19,8,20,17,5,13,6,12,16,11,22,18,14,15)
Nug24	24	3488		(OPT) (17,8,11,23,4,20,15,19,22,18,3,14,1,10,7,9,16,21,24,12,6,13,5,2)
Nug25	25	3744		(OPT) (5,11,20,15,22,2,25,8,9,1,18,16,3,6,19,24,21,14,7,10,17,12,4,23,13)
Nug27	27	5234		(OPT) (23,18,3,1,27,17,5,12,7,15,4,26,8,19,20,2,24,21,14,10,9,13,22,25,6,16,11)
Nug28	28	5166		(OPT) (18,21,9,1,28,20,11,3,13,12,10,19,14,22,15,2,25,16,4,23,7,17,24,26,5,27,8,6)
Nug30	30	6124		(OPT) (14,5,28,24,1,3,16,15,10,9,21,2,4,29,25,22,13,26,17,30,6,20,19,8,18,7,27,12,11,23)

B.7: Skorin-Kapov QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Sko42	42	15812	14934	5.56 %
Sko49	49	23386	22004	5.91 %
Sko56	56	34458	32610	5.37 %
Sko64	64	48498	45736	5.70 %
Sko72	72	66256	62691	5.38 %
Sko81	81	90998	86072	5.41 %
Sko90	90	115534	108493	6.10 %
Sko100a	100	152002	142668	6.14 %
Sko100b	100	153890	143872	6.51 %
Sko100c	100	147862	139402	5.73 %
Sko100d	100	149576	139898	6.47 %
Sko100e	100	149150	140105	6.07 %
Sko100f	100	149036	139452	6.43 %

B.8: Taillard QAPLIB Files

Name	N	Feasible Solution	Permutation/Bound	Gap
Tai12a	12	224416 (OPT)	(8,1,6,2,11,10,3,5,9,7,12,4)	
Tai12b	12	39464925 (OPT)	(9,4,6,3,11,7,12,2,8,10,1,5)	
Tai15a	15	388214 (OPT)	(5,10,4,13,2,9,1,11,12,14,7,15,3,8,6)	
Tai15b	15	51765268 (OPT)	(1,9,4,6,8,15,7,11,3,5,2,14,13,12,10)	
Tai17a	17	491812 (OPT)	(12,2,6,7,4,8,14,5,11,3,16,13,17,9,1,10,15)	
Tai20a	20	703482 (OPT)	(10,9,12,20,19,3,14,6,17,11,5,7,15,16,18,2,4,8,13,1)	
Tai20b	20	122455319 (OPT)	(8,16,14,17,4,11,3,19,7,9,1,15,6,13,10,2,5,20,18,12)	
Tai25a	25	1167256	1016213	12.94 %
Tai25b	25	344355646 (OPT)	(4,15,10,9,13,5,25,19,7,3,17,6,18,20,16,2,22,23,8,11,21, 24,14,12,1)	
Tai30a	30	1818146	1529135	15.90 %
Tai30b	30	637117113	40947945	93.58 %
Tai35a	35	2422002	1951207	19.44 %
Tai35b	35	283315445	32611838	88.49 %
Tai40a	40	3139370	2492850	20.60 %
Tai40b	40	637250948	46143753	92.77 %
Tai50a	50	4941410	3854359	22.00 %
Tai50b	50	458821517	40296004	91.23 %
Tai60a	60	7208572	5555095	22.94 %
Tai60b	60	608215054	50113782	91.77 %
Tai64c	64	1855928	896398	51.71 %
Tai80a	80	13557864	10329674	23.82 %
Tai80b	80	818415043	89169828	89.11 %
Tai100a	100	21125314	15824355	25.10 %
Tai100b	100	1185996137	174687926	86.28 %
Tai150b	150	498896643	63007151	87.37 %
Tai256c	256	44759294	41291996	7.75 %