DESIGN OF PERTURBATIVE HYPER-HEURISTICS FOR COMBINATORIAL OPTIMISATION

by

CHOONG SHIN SIANG

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

June 2019

ACKNOWLEDGEMENT

I am heartily thankful to my main supervisor, Dr. Wong Li Pei, whose encouragement, guidance and support from the initial to the final level of this research, enabled me to develop an understanding of the subject. Dr. Wong demonstrated his passion and enthusiasm in the supervision process. He provided me tonnes of suggestions, ideas and criticism throughout my entire candidature period. Special thanks to my co-supervisor, Professor Lim Chee Peng who guided me in the exploration of previously unfamiliar research areas. His insightful guidance is very helpful in conducting and publishing the research.

Not to forget my thanks to my parents (Choong Moi Kooi and Kuan Kar Foong) for being supportive during the period of my study. I am also grateful to Ms. Liew Yann Wen who helped me in proofreading the thesis before the final submission.

Besides that, I acknowledge Universiti Sains Malysia Graduate Assistant Scheme for its financial support. By working under this scheme, I have gained precious teaching experiences and knowledge in addition to the allowance. Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
ACO	Ant Colony Optimisation
AM	All Moves
APW	Accept Probabilistic Worse
BA	Bees Algorithm
BCO	Bee Colony Optimisation
BPP	Bin-Packing Problem
BS	Bee System
CALO	Chaotic AntLion Optimization
CHeSC	Cross-domain Heuristic Search Challenge
CLK	Chained Lin-Kernighan
CSA	Crow search algorithm
DT_2-opt	Delaunay Triangulation based 2-opt
FCCD	Face-centred Central Composite Design
GA	Genetic Algorithm
GEP	Gene Expression Programming
GOA	Grasshopper Optimization Algorithm
GP	Genetic Programming
HBMO	Honey Bees Mating Optimisation
HM	Harmony Memory
HMM	Hidden Markov Model
HyFlex	Hyper-heuristic Flexible Framework

ILS	Iterated Local Search
IWD	Intelligence Water Drop
KP	Knapsack Problem
LA	Late Acceptance
LK	Lin-Kernighan
LLH	low-level heuristic
MA	Metropolis Acceptance
MAB	Multi-Armed Bandit
MCF	Modified Choice Function
MCF-ABC	A model integrated MCF in ABC
MCTS	Monte Carlo Tree Search
NA	Naive Acceptance
OI	Only Improvement
PFSP	Permutation Flow Shop Problem
POMDP	Partially Observable Markov Decision Process
PSO	Particle Swarm Optimisation
PSP	Personnel Scheduling Problem
QHH	A Q-learning based model to automatically design hyper-heuristics
RI	Random Insertion
RIS	Random Insertion of Subsequence
RL	Reinforcement Learning
RRIS	Random Reversing Insertion of Subsequence
RRS	Random Reversing of Subsequences
RRSS	Random Reversing Swap of Subsequences
RS	Random Swap

RSIS	Random Shuffle Insertion of Subsequence
RSS	Random Swap of Subsequences
RSSS	Random Shuffle Swap of Subsequence
SA	Simulated Annealing
SAT	Boolean Satisfiability Problem
SS	Shuffle Subsequence
SSA	Salp Swarm Algorithm
TSP	Traveling Salesman Problem
TTP	Traveling Thief Problem
VRP	Vehicle Routing Problem

REKA BENTUK HIPER-HEURISTIK USIK UNTUK PENGOPTIMUMAN KOMBINATORIK

ABSTRAK

Pengoptimuman kombinatorik ialah suatu bidang yang bertujuan untuk mengenal pasti penyelesaian optimum daripada sesuatu ruang gelintaran penyelesaian diskret. Cara-cara untuk menyelesaikan masalah pengoptimuman kombinatorik boleh dibahagikan kepada dua subkelas utama, iaitu algoritma tepat dan algoritma penghampiran. Algoritma tepat ialah satu teknik yang menjamin pengoptimuman global. Akan tetapi, algoritma tepat tidak dapat digunakan untuk menyelesaikan masalah yang rumit kerana overhed perkomputerannya yang tinggi. Algoritma penghampiran ialah satu teknik yang boleh membekalkan penyelesaian sub-optimum dengan kos perkomputeran yang munasabah. Untuk meneroka ruang gelintaran penyelesaian bagi sesuatu masalah pengoptimuman kombinatorik, algoritma penghampiran menjalankan usikan terhadap penyelesaian yang sedia ada dengan menggunakan satu atau pelbagai heuristik usik peringkat rendah. Penggunaan satu dan hanya satu heuristik peringkat rendah menyebabkan prestasi yang buruk apabila heuristik tersebut tidak cekap untuk menyelesaikan masalah itu. Oleh itu, penggunaan pelbagai heuristik peringkat rendah lebih wajar kerana kelemahan satu heuristik dapat diimbangi oleh kelebihan heuristik yang lain. Apabila pelbagai heuristik peringkat rendah digunakan, hiper-heuristik boleh diintegrasikan untuk memilih heuristik yang sesuai dengan satu masalah atau situasi. Hiper-heuristik mengautomatikkan pemilihan heuristic-heuristik peringkat rendah melalui satu heuristik peringkat tinggi yang terdiri daripada dua komponen utama, iaitu satu kaedah pemilihan heuristik dan satu kaedah penerimaan penyelesaian. Keupayaan satu heuristik peringkat tinggi bergantung kepada sesuatu masalah kerana landskap sesuatu masalah adalah unik antara satu dengan yang lain. Heuristik peringkat tinggi dalam hiper-heuristik yang sedia ada direka bentuk dengan memadankan pelbagai kombinasi komponen heuristik peringkat tinggi secara manual. Tujuan reka bentuk yang manual itu adalah menentukan satu heuristik peringkat tinggi yang terbaik untuk sesuatu masalah. Akan tetapi, percubaan tuntas mengambil masa yang lama kerana ia melibatkan pemilihan reka bentuk yang banyak. Oleh itu, satu kaedah mereka bentuk heuristik peringkat tinggi secara automatik yang efektif diperlukan untuk membangun satu hiper-heuristik yang berkesan. Penyelidikan ini mengintegrasikan satu hiper-heuristik, iaitu Modified Choice Function (MCF) untuk mengawal pemilihan heuristik peringkat rendah dalam satu algoritma penghampiran, iaitu algoritma Artificial Bee Colony (ABC). Khususnya, MCF digunakan untuk menentukan heuristik peringkat rendah yang sesuai dengan setiap operasi gelintaran kejiranan dalam proses pengoptimuman. Model yang dicadangkan ini bernama MCF-ABC. MCF-ABC menyelesaikan 64 kes Masalah Perjalanan Jurujual hingga 0.055% dari optimum dalam masa 2.7 minit. Kajian perbandingan antara algoritma-algoritma state-of-the-art membuktikan keberkesanan MCF-ABC. Seterusnya, penyelidikan ini mencadangkan satu model untuk mereka bentuk heuristik peringkat tinggi bagi hiper-heuristik secara automatik dengan menggunakan suatu algoritma Reinforcement Learning (RL), iaitu Q-learning. Dalam model yang dicadangkan ini, RL digunakan untuk membimbing pemilihan komponen heuristik peringkat tinggi yang sesuai untuk pelbagai fasa dalam proses pengoptimuman. Model yang dicadangkan ini bernama QHH. Apabila dibandingkan dengan 33 hyper-heuristik state-of-the-art, QHH berada pada kedudukan keenam, yang adalah dalam persentil ke-18. Selain itu, satu kajian kes dunia sebenar yang berkaitan dengan pemilihan ciri dalam pengelasan dijalankan untuk menunjukkan kebolehgunaan model-model yang dicadangkan. Dalam kajian kes tersebut, model-model yang dicadangkan digunakan untuk melaksanakan pemilihan ciri dalam pengesanan gerakan manusia. Keputusan empirikal menunjukkan bahawa MCF-ABC dan QHH dapat mengenal pasti subset-subset ciri yang berguna untuk pengelasan. Keputusan-keputusan tersebut membuktikan model-model yang dicadangkan berkeupayaan untuk menyelesaikan masalah-masalah tanda aras dan dunia sebenar.

DESIGN OF PERTURBATIVE HYPER-HEURISTICS FOR COMBINATORIAL OPTIMISATION

ABSTRACT

Combinatorial optimisation is an area which seeks to identify optimal solution(s) from a discrete solution search space. Approaches for solving combinatorial optimisation problems can be separated into two main sub-classes, i.e. exact and approximation algorithms. Exact algorithm is a sub-class of techniques that is able to guarantee global optimality. However, exact algorithms are not feasible for solving complex problem due to its high computational overhead. Approximation algorithm is a sub-class of techniques which is able to provide sub-optimal solution(s) with reasonable computational cost. In order to explore the solution search space of a combinatorial optimisation problem, an approximation algorithm performs perturbations on the existing solutions by adopting a single or multiple perturbative Low-Level Heuristic(s) (LLHs). The use of a single LLH leads to poor performance when the particular heuristic is incompetent in solving the problem. Thus, the use of multiple LLHs is more desirable as the weaknesses of one heuristic can be compensated by the strengths of another. When there are multiple LLHs, a hyper-heuristic can be integrated to determine the choice of heuristics for a particular problem or situation. Hyper-heuristic automates the selection of LLHs through a high-level heuristic that consists of two key components, i.e. a heuristic selection method and a move acceptance method. The capability of a high-level heuristic is highly problem dependent as the landscape properties of a problem are unique among others. The high-level heuristics in the existing hyper-heuristics are designed by manually matching different combinations of high-level heuristic components. In such manual design, the concern is to determine the best high-level heuristic for a particular problem. However, exhaustive attempts are time consuming due to the large availability of design choices. Hence, an effective approach for automatically designing the high-level heuristic is necessary to develop a robust hyper-heuristic. This research first integrates a hyper-heuristic, namely Modified Choice Function (MCF) to regulate the selection of low-level heuristics in an approximation algorithm. Specifically, MCF is employed to determine an appropriate low-level heuristic for each neighbourhood search operation in the optimisation process. This proposed model is named as MCF-ABC. MCF-ABC solves 64 Traveling Salesman Problem (TSP) instances to 0.055% from the known optimum within 2.7 minutes. Comparison studies among the state-of-the-art algorithms prove the competitiveness of MCF-ABC. Next, a model is proposed to automatically design the high-level heuristic of a hyper-heuristic by utilising a Reinforcement Learning algorithm, namely Q-learning. In the proposed model, Q-learning is applied to guide the hyper-heuristic in selecting the proper high-level heuristic components for different stages of the optimisation process. This proposed model is named as QHH. When QHH is compared against 33 state-of-the-art hyper-heuristics, it is ranked at the sixth position, which is within the 18th percentile. Besides that, a real-world case study related to feature selection for classification is conducted to further demonstrate the applicability of the proposed models. In this case study, the proposed models are applied to perform feature selection for human motion detection. The empirical results show that MCF-ABC and QHH are able to identify useful feature subset for the classification task. These results ascertain the usefulness of the proposed models in solving both the benchmark and real-world problems.

CHAPTER 1

INTRODUCTION

Combinatorial optimisation (Papadimitriou & Steiglitz, 1998) is one of the areas that has been studied intensively in computer science and operations research. Solving a combinatorial optimisation problem involves finding the feasible solutions from a finite set of solutions that exist in a discrete solution search space, and then identifying only the optimal solution(s). The examples of combinatorial optimisation problems include Boolean Satisfiability Problem (SAT) (Vardi, 2014, Xu et al., 2019), Bin-Packing Problem (BPP) (Schwerin & Wäscher, 1997, Bertazzi et al., 2019), Permutation Flow Shop Problem (PFSP) (Framinan et al., 2004, Almeida et al., 2018), Personnel Scheduling Problem (PSP) (Porto et al., 2019), Traveling Salesman Problem (TSP) (Laporte, 1992a, Lancia & Serafini, 2018), Vehicle Routing Problem (VRP) (Laporte, 1992b, Nazari et al., 2018), Traveling Thief Problem (TTP) (Bonyadi et al., 2013, Yafrani et al., 2017), etc.

Approaches for solving combinatorial optimisation problems can be categorised into two main groups, i.e. exact and approximation algorithms. Exact algorithms (Woeginger, 2003) are deterministic. In other words, multiple runs of an exact algorithm always result in the generation of a same solution. Exact algorithms utilise mathematical models in its problem solving process. One of the major advantages of the exact algorithms is that it guarantees global optimality. However, the guarantee is limited to small problems. For complex problems (i.e. problems with complicated relation among the candidate solutions and their qualities), an exact algorithm may take a long time to reach optimality (Talbi, 2009). Thus, exact algorithms are inefficient for solving NP-hard problems with large dimension (Hochbaum, 1996).

To solve this kind of complex problems, the application of approximation algorithms (Johnson, 1974) to produce solutions that are good enough for solving the problem in a reasonable time frame is a viable option. Approximation algorithms make use of iterative improvements in its problem solving process. Due to the existence of randomness in approximation algorithms, different solutions can be obtained in different runs of a same approximation algorithm. Approximation algorithms usually make use of heuristics to generate or improve solutions.

A heuristic involves applying a practical methodology that does not guarantee convergence to a global optimum, but is able to provide a sufficiently good solution. For solving a combinatorial optimisation problem, two types of heuristics are available, i.e. perturbative heuristic and constructive heuristic (Burke et al., 2010b). A constructive heuristic incrementally builds a complete solution from scratch. On the other hand, a perturbative heuristic modifies an existing solution in order to look for better solutions in the neighbourhood of the existing solution (i.e. operates neighbourhood search operations). Heuristics are problem specific. For example, the constructive heuristics for TSP include the nearest neighbourhood heuristic and the multiple fragment heuristic, while the perturbative heuristics for TSP include swap mutation, insert mutation, reverse mutation, order crossover, and partially mapped crossover.

An approximation algorithm can employ a single or multiple heuristic(s) in its optimisation process. Examples of approximation algorithms which employ a constructive heuristic include Ant Colony Optimisation (ACO) (Dorigo et al., 2006, Dorigo & Stützle, 2019) and Bee Colony Optimisation (BCO) (Wong, 2012, Choo et al., 2016). Examples of approximation algorithms which employ perturbative heuristics include Genetic Algorithm (GA) (Holland, 1992, Mirjalili, 2019a), Iterated Local Search (ILS) (Stützle & Ruiz, 2017), Simulated Annealing (SA) (Van Laarhoven & Aarts, 1987, Chopard & Tomassini, 2018), Particle Swarm Optimisation (PSO) (Kennedy, 2011, Mirjalili, 2019b), and Artificial Bee Colony (ABC) (Karaboga & Gorkemli, 2011, Hussain et al., 2018) algorithms.

As there is a large variety of available problem specific heuristics, the key question concerning the selection of a particular heuristic has been posed in the literature in recent years. This question formulates a heuristic search space, which contains a number of possible arrangements on when to use which heuristic in the optimisation process. This leads to the studies on hyper-heuristics, which is the focus of this research. Instead of directly operating search on the solution search space, hyper-heuristic performs searching on the heuristic search space.

1.1 Hyper-heuristic

A hyper-heuristic is a high-level automated methodology for selecting or generating a set of heuristics (Burke et al., 2013). The term “hyper-heuristic” was coined by Denzinger et al. (1996). Figure 1.1 shows a classification of hyper-heuristic approaches. There are two main hyper-heuristic categories, i.e. selection hyper-heuristic and generation hyper-heuristic (Burke et al., 2010b). These two categories can be defined as 'heuristics to select heuristics' and 'heuristics to generate heuristics' respectively (Burke et al., 2013). The heuristics to be selected or generated in a hyper-heuristic are known as the low-level heuristics (LLHs). Both selection and generation hyper-heuristics can be further divided into two categories based on the nature of the LLHs (Burke et al., 2010b), namely either constructive or perturbative hyper-heuristics. A constructive hyper-heuristic incrementally builds a complete solution from scratch by selecting or generating constructive LLHs. On the other hand, a perturbative hyper-heuristic iteratively improves an existing solution by performing perturbations on the existing candidate solutions using perturbative LLHs. According to Ochoa et al. (2012), perturbative LLHs can be categorised into ruin-recreate heuristics, mutation heuristics, crossover heuristics, and hill-climbing heuristics.

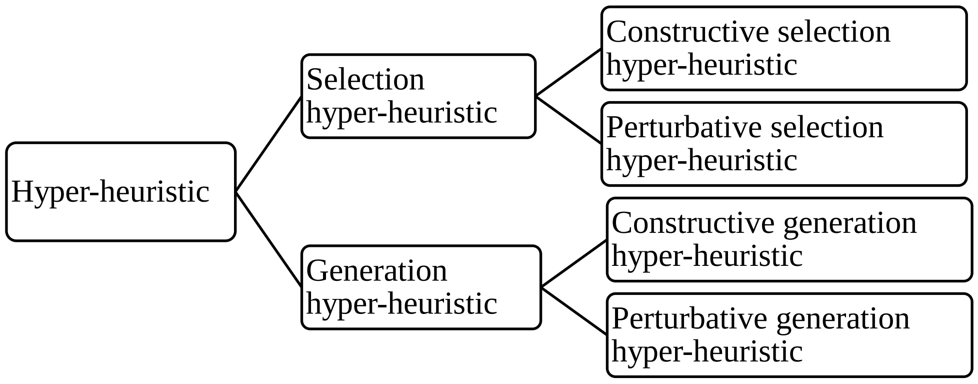

Hyper-heuristic
Selection hyper-heuristic
Generation hyper-heuristic
Constructive selection hyper-heuristic
Perturbative selection hyper-heuristic
Constructive generation hyper-heuristic
Perturbative generation hyper-heuristic

Figure 1.1 A classification of hyper-heuristics

In addition, hyper-heuristic can also be classified based on the number of solutions maintained in the memory. A single-point based hyper-heuristic maintains only a single solution, while a multi-point based hyper-heuristic maintains a population of solutions in its memory. A single-point based selection hyper-heuristic consists of two levels, as shown in Figure 1.2 (Burke et al., 2013). The low level contains a representation of the problem, evaluation function(s), and a set of problem specific LLHs. The high level manages which LLH to use for producing a new solution(s), and then decides whether to accept the solution(s). Therefore, the high-level heuristic performs two separate tasks i.e. (i) LLH selection and (ii) move acceptance (Ö zcan et al., 2008). The LLH selection method is a strategy to select an appropriate perturbative LLH to modify the current solution and the move acceptance method decides whether to accept the newly generated solution. A multi-point based hyper-heuristic has similar low-level and high-level components to a single-point hyper-heuristic. However, the high level of a multi-point based hyper-heuristic includes a memory mechanism, which controls the storage, selection, and replacement of the solutions in the memory.

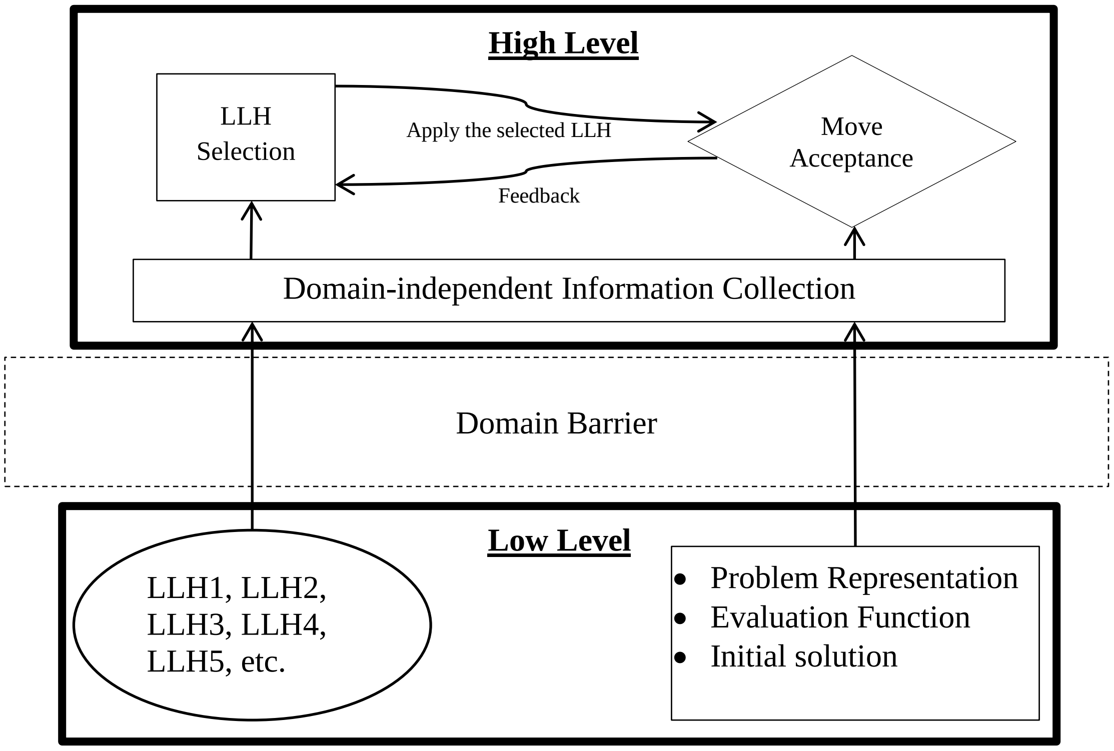


Figure 1.2 A generic structure of a single-point based hyper-heuristic model (Burke et al., 2010b)

In the optimisation process of a hyper-heuristic, exploration and exploitation are performed in two different search spaces, i.e. a solution search space and an LLH search space. The solution search space contains a set of candidate solutions to solve the combinatorial optimisation problem, while the heuristic search space consist of a set of feasible arrangements on the use of the available heuristic.

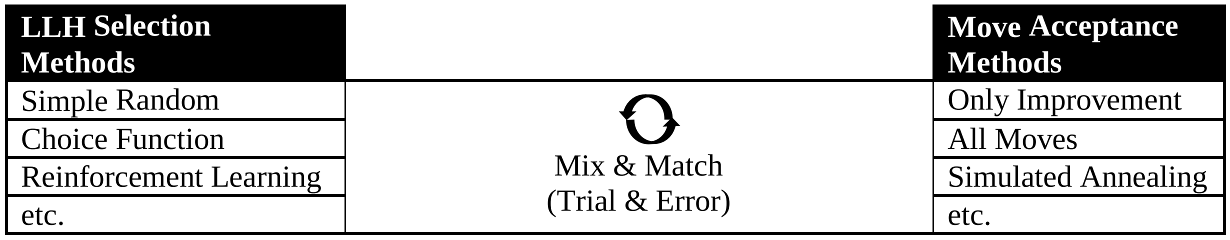
1.2 Problem Statement and Research Questions

When approximation algorithms, such as SA, ILS, GA, and ABC are applied to solve a combinatorial optimisation problem, perturbative LLH(s) is adopted to explore the solution search space. Some approximation algorithms statically adopt a single perturbative LLH throughout the optimisation process. For instance, an GA for solving TSP only applies the insertion mutation as mutation operator for the whole optimisation process without considering other mutation heuristics such as swap mutation or inversion mutation (Fogel, 1988). This results in low performance when the particular LLH is not suitable for the problem being solved. Besides that, different LLHs may have different capabilities in handling different situations during the optimisation process. For example, an explorative LLH is useful during the earlier stages of the optimisation, while an exploitative LLH is capable in the converging stages. To compensate the weaknesses of one heuristic by the strengths of another, multiple perturbative LLHs can be employed in an approximation algorithm. This rises the first research question in this study:

When there are multiple perturbative LLHs, how to determine the selection of LLHs at different stages of the optimization process?

The selection of LLHs formulates an LLH search space. A mechanism (i.e. a hyper-heuristic) to operate search on the LLH search space (i.e. to determine when to use which LLH) is necessary to enhance the performance of an approximation algorithm (Yan & Wu, 2015, Castro et al., 2018).

Hyper-heuristic is an automated methodology for selecting LLHs. It accomplishes this task through a high-level heuristic (i.e. an LLH selection method & a move acceptance method). Both LLH selection and move acceptance methods have a dramatic impact on the performance of a hyper-heuristic. Their effects are problem-dependent as different problem domains, or even different instances in a single domain, have different fitness landscape properties (Sabar et al., 2015a). Choosing suitable LLH selection and move acceptance methods for a particular problem is a non-trivial task during the process of designing a robust hyper-heuristic. As there are many different pairwise combinations of LLH selection and move acceptance methods, the high-level heuristic space is large (Burke et al., 2013). In most of the existing perturbative hyper-heuristics, the high-level heuristics are designed manually (Sabar et al., 2015a). The aim of such manual design is to determine which combination of the LLH selection and move acceptance methods is the best for a particular problem. One of the most straightforward methods is to find the best combination for each problem using a trial-and-error approach. However, this is extremely time-consuming as the possible number of combinations of these methods is large. In addition, the optimal configuration of these methods varies during different stages of the optimisation process. Figure 1.3 illustrates the limitations of manually designing perturbative hyper-heuristics.



Disadvantages of the Mix & Match strategy:

- Time-consuming
- The optimal configuration of these methods varies during different stages of the optimisation process.

Can the design process be automated?

Figure 1.3 Limitations of manually designing perturbative hyper-heuristics

To overcome the limitations of manual design process, methods for automatically designing perturbative hyper-heuristics have been proposed. The method to automatically design perturbative hyper-heuristics can be categorised in a similar way as those of hyper-heuristics, i.e. automatic selection and automatic generation of high-level heuristic components. Both categories can be further divided as online-design and offline-design methods. In the case of online-design, the design process takes place during the real optimisation process, i.e. when a problem instance is being solved. If the design process is accomplished before the real optimisation process (i.e. by solving a set of training instances), it is considered as an offline-design method. Offline-selection (Adriaensen et al., 2014a) and online-generation (Sabar et al., 2015a, Fontoura et al., 2017) methods have been employed by the existing studies on automatic design of hyper-heuristics. Sabar et al. (2015a) and Fontoura et al. (2017) applied online generation methods that utilised Genetic Programming (GP) based rule generation strategies. One drawback of these GP based strategies is the search space size grows exponentially in both the number of components that form a rule and the number of generated rules (Chen et al., 2015). Adriaensen et al. (2014a) utilised an offline-selection method. The drawback of this offline method is that it is highly dependent on the generality of the training instances. To generalise the performance of the designed perturbative hyper-heuristics to other problems, the set of training instances need to be sufficiently large. However, a large set of training instances highly increases the computational cost of the design process. The detail descriptions about Sabar et al. (2015a), Fontoura et al. (2017) and Adriaensen et al. (2014a) are provided in Section 2.5. These issues lead to the following research questions:

Can the design process of a perturbative hyper-heuristic be automated with a method that compensates the weaknesses of the two existing approaches (i.e. does not require an offline training process like the offline-selection method, and involves a smaller search space as compared with the online-generation method)?

1.3 Research Objectives

The ultimate goal of this research is to solve combinatorial optimisation problems using perturbative hyper-heuristics. The combinatorial optimisation problems that are used as test bed of this study include TSP, TTP, SAT, BPP, PFSP, PSP, VRP, and feature selection. To address the problems stated in Section 1.2, the objectives of this research are set as follows:

1. To integrate a perturbative hyper-heuristic in an approximation algorithm for regulating the selection of perturbative LLHs.

2. To propose a model for automatically designing robust perturbative hyper-heuristics by selecting high-level heuristic components during the optimisation process (i.e. online-selection).
3. To evaluate the proposed models on a real-world optimisation problem related to feature selection for classification.

The first objective of this research is to examine the effect of integrating a perturbative hyper-heuristic to improve the performance of an approximation algorithm pertaining to combinatorial optimisation. In the experiments, a perturbative hyper-heuristic namely, Modified Choice Function (MCF) is used to guide the selection of perturbative LLHs used for each neighbourhood search during the optimisation process of an approximation algorithm, namely the Artificial Bee Colony (ABC) algorithm. The proposed model is denoted as MCF-ABC.

The second objective of this research is to propose an online-selection method for automatically designing robust perturbative hyper-heuristics, i.e. a method to select appropriate high-level heuristic components during the optimisation process. The proposed model compensates the drawbacks of the existing approaches, i.e. the online-generation method (Sabar et al., 2015a, Fontoura et al., 2017) and the offline-selection methods (Adriaensen et al., 2014a). The proposed model does not require an offline training process like the offline-selection method, and at the same time, leads to a reduction of the search space as compared with the online-generation method. A Reinforcement Learning (RL) algorithm, namely Q-learning is used such that the proposed model is able to intelligently select suitable high-level heuristic components by learning from experience during the optimisation process. The resulting Q-learning based hyper-heuristic model is denoted as QHH.

The performance of the proposed models is assessed using a set of benchmark combinatorial optimisation problem instances. Besides that, the effectiveness of the proposed models is examined using a feature selection for classification problem. First, the potentials of the proposed QHH and MCF-ABC models are evaluated using a set of benchmark datasets obtained from the UCI machine learning repository (Lichman, 2013). Next, the proposed models are applied to perform feature selection for a real-world human motion detection and classification task.

1.4 Research Methodology

This research includes three major phases. The first phase integrates a perturbative hyper-heuristic in an approximation algorithm. Specifically, the hyper-heuristic is applied to select suitable perturbative LLHs for the approximation algorithm. The proposed model and research findings of this phase are discussed in Chapter 3.

The second phase proposes a model to automatically design perturbative hyper-heuristics. The proposed model is able to select appropriate high-level heuristic components for different stages of the optimisation process. This proposed model and its evaluation results are reported in Chapter 4.

In the third phase, the proposed models are assessed using a real-world problem related to feature selection for classification. The details of this phase are described in Chapter 5. Figure 1.4 summaries the three phases of this research.

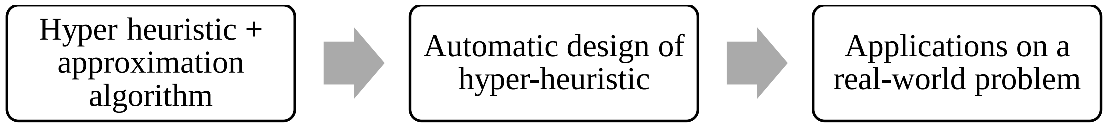

Hyper heuristic + approximation algorithm
Automatic design of hyper-heuristic
Applications on a real-world problem

Figure 1.4 Three main phases in this research

For each of these phases, a step-by-step methodological procedure is implemented in order to accomplish the proposed research objectives. The methodological procedure is summarised in Figure 1.5. First, a problem analysis which includes background studies, research gap identification, problem modelling, and data collection is performed. Next, the design and development of models is initiated with a pseudo code generation, and followed by implementation of proposed models and parameter tuning using a structured design-of-experiment technique, namely the Face-centred Central Composite Design (FCCD). Then, the proposed models are tested on a set of benchmark combinatorial problem instances and a real-world problem related to feature selection for classification. All experiments were conducted using a workstation with multiple Intel i7-3930K 3.20 GHz processors, and with 15.6GB of memory. At any particular time, each test was executed by one processor only. A statistical test, namely Wilcoxon Signed-Rank test (Wilcoxon et al., 1970) is conducted to statistically compare the performance of the proposed models with the state-of-the-arts approaches. The results are compiled as publications in international journals and conference proceedings, as well as reported in this thesis.

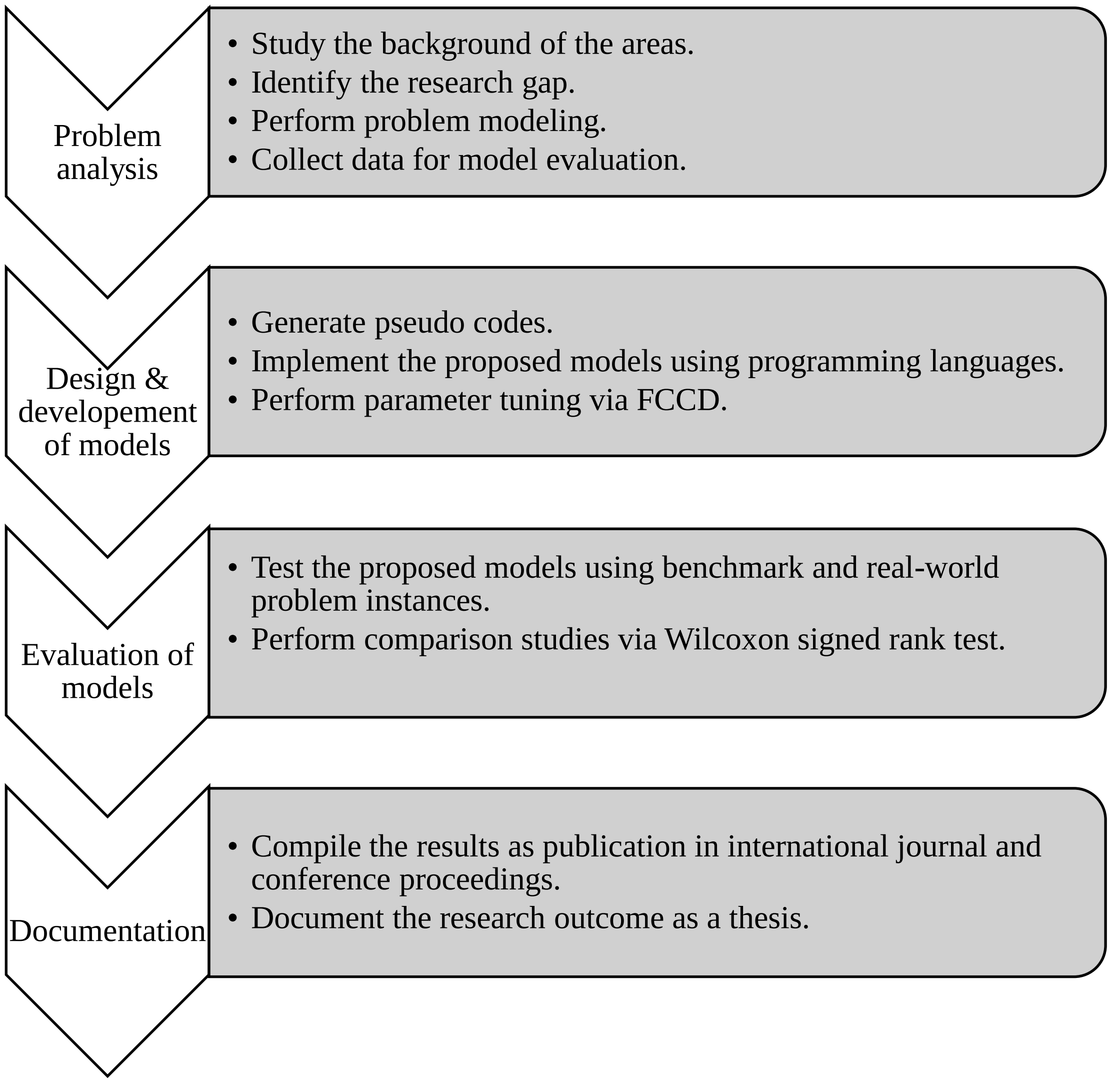

Problem analysis
• Study the background of the areas.
• Identify the research gap.
• Perform problem modeling.
• Collect data for model evaluation.
Design & developement of models
• Generate pseudo codes.
• Implement the proposed models using programming languages.
• Perform parameter tuning via FCCD.
Evaluation of models
• Test the proposed models using benchmark and real-world problem instances.
• Perform comparison studies via Wilcoxon signed rank test.
Documentation
• Compile the results as publication in international journal and conference proceedings.
• Document the research outcome as a thesis.

Figure 1.5 The methodological procedure of this research

1.5 Research Scope

The two proposed models in this research (i.e. MCF-ABC and QHH) are categorised as online-selection methods that do not require an offline training process. Specifically, the proposed MCF-ABC models select LLHs during the optimisation process, while QHH selects combinations of high-level heuristic components during the optimisation process. Furthermore, the LLHs to be selected are all perturbative heuristics. The constructive heuristics are not included in the scope of the study.

To achieve the first objective of this study (i.e. to integrate a perturbative hyper-heuristic in an approximation algorithm for regulating the selection of perturbative LLHs), experiments are conducted based on a perturbative hyper-heuristic known as MCF (Drake et al., 2012) and an approximation algorithm, namely ABC. The reasons of choosing ABC and MCF in the experiments are as follows. ABC has a unique neighbourhood search mechanism for solving optimisation of mathematical test functions with continuous search spaces (Karaboga, 2005). Unfortunately, the neighbourhood search mechanism cannot be directly applied in the discrete version of ABC for solving combinatorial optimisation problems with discrete search spaces. Instead, perturbative LLH(s) are adopted to perform the neighbourhood search. However, there are limited studies concerning the LLH selection of ABC in the literature. This leads to the motivation of integrating a hyper-heuristic to regulate the selection of LLHs in the discrete version of ABC to improve its performance on a combinatorial optimisation problem. MCF is chosen among the available hyper-heuristics because it is able to adaptively control the weights of its intensification and diversification components during different stages of the optimisation process.

Related to the second objective (i.e. to propose a model for automatically designing robust perturbative hyper-heuristics by selecting high-level heuristic components during the optimisation process), the definition of the high-level heuristic is based on the generic structure of a single-point based hyper-heuristic model (Burke et al., 2013) as shown in Figure 1.2. It consists of two components, i.e. an LLH selection method and a move acceptance method. The proposed model to automatically select the high-level heuristics can only be applied on the hyper-heuristics that have these two components. In the proposed model, the design process of a high-level heuristic draws certain properties pertaining to a Partially Observable Markov Decision Process (POMDP), which the details can be found in Section 4.1. Based on previous researches, POMDPs can be effectively solved using an RL technique, namely Q-learning (Lee, 2000, Karadeniz & Akin, 2004, Liu et al., 2007). Therefore, Q-learning is used in the proposed model such that it is able to learn from experience by interacting with the environment.

The applicability of the proposed models is evaluated using a real-world optimisation problem related to feature selection for classification. Feature selection is important in various areas including pattern recognition (Jensen & Shen, 2009, Diao & Shen, 2012), gene expression array analysis (Maji & Paul, 2011), and text mining (Aghdam et al., 2009). It can be modelled as a combinatorial optimisation problem which aims to identify optimal feature subsets that are relevant for a classification task. Therefore, feature selection for classification is an appropriate domain to examine the effectiveness of the proposed models. In this study, the proposed models are applied to perform feature selection on a real-world human motion detection dataset. The significance of human motion detection and classification can be found in various types of applications, e.g. clinical management, fraud detection, and healthcare (Bayat et al., 2014).

1.6 Research Contributions

The research objectives mentioned in Section 1.3 leads to three tangible contributions. The first contribution of this research is to integrate a perturbative hyper-heuristic, i.e. MCF (Drake et al., 2012) in an approximation algorithm, i.e. ABC. MCF is employed to guide the selection of the perturbative LLHs in the optimisation process, such that suitable LLH is selected for each neighbour search. The proposed model is denoted as MCF-ABC. MCF-ABC is tested on a set of TSP and TTP instances. The performance comparison with other state-of-the-art approaches shows the effectiveness of the proposed MCF-ABC model.

Besides that, a model to automatically design the high-level heuristics of a perturbative hyper-heuristic is proposed. An RL technique, i.e. Q-learning is applied to intelligently select appropriate high-level heuristic components (i.e. LLH selection-move acceptance pairs) for different stages of the optimisation process. The proposed model is denoted as QHH. As mentioned in Section 1.3, QHH is an online selection method that involves a reasonable high-level heuristic search space and does not require a time-consuming offline training process. QHH is evaluated using benchmark instances from six problem domains in the Hyper-heuristic Flexible Framework (HyFlex). The experimental results show that performance of QHH is comparable with those of the top-performing hyper-heuristics in the literature.

In addition, the proposed MCF-ABC and QHH models are applied to perform feature selection for classification. The proposed models are first evaluated on a set of benchmark UCI datasets. In the comparison with other state-of-the-art feature selection approaches, MCF-ABC and QHH obtain competitive results. Next, the proposed models are used to perform feature selection on a real-world human motion dataset. The experimental results show that the proposed models are able to accomplish good classification accuracies with reduced feature subsets. This indicates that the proposed MCF-ABC and QHH models are capable in identifying the useful features for human motion detection and classification.

1.7 Thesis Outline

This thesis consists of six chapters. It begins with this introduction chapter that presents an overview of the research. It highlights the background, problem statements, objectives, methodology, scope, and contributions of this research.

Chapter 2 reviews some related work of the study. First, a review on various hyper-heuristics and approximation algorithms are provided. Then, the integration of hyper-heuristics in approximation algorithms is discussed. Next, applications of RL in hyper-heuristics are presented. Finally, existing approaches to automatically design hyper-heuristics are reviewed.

Chapter 3 presents a proposed model which integrates a hyper-heuristic (i.e. MCF) in an approximation algorithm (i.e. ABC). It includes an explanation of MCF, a pseudo code of the proposed model and applications of the proposed MCF-ABC model to two benchmark combinatorial optimisation problems, i.e. TSP and TTP. For each of the applications, an introduction of the problem (i.e. TSP and TTP), a description of LLHs, and a picture of the experimental results are provided.

Chapter 4 presents a proposed model to automatically design the high-level heuristic of a hyper-heuristic. The automatic design process of high-level heuristics is modelled as a POMDP. An RL technique, namely Q-learning is adapted to intelligently select proper high-level heuristic components for different stages of the optimisation process. After an overview of Q-learning are presented, the details of the proposed QHH model such as the pseudo code, action and state representations, reinforcement signal, Q-function parameters, execution of the designed hyper-heuristics, are discussed. Next, a numerical example is provided and the experimental studies are presented.

Chapter 5 describes the application of the two proposed models on a real-world optimisation problem related to feature selection for classification. First, the significance and formulation of the problem is described. Then, the adaptation of the proposed models to solve the feature selection problem is explained. Finally, the experimental results are revealed.

Chapter 6 provides a conclusion of the research. Concluding remarks and a summary of the key findings in this research are highlighted in this chapter. Besides that, a number of potential future research directions are suggested.

CHAPTER 2

LITERATURE REVIEW

This chapter describes the related work of the research. Section 2.1 describes combinatorial optimisation. Section 2.2 and Section 2.3 presents a description of hyper-heuristic and approximation algorithm respectively. Section 2.4 provides a review on the integration of hyper-heuristics in approximation algorithms. Section 2.5 describes the applications of RL in hyper-heuristic. Section 2.6 reviews the existing approaches to automatically design hyper-heuristics. Section 2.7 summaries the important points reviewed in this chapter. Besides that, an analysis of the research gap is conducted.

2.1 Combinatorial Optimisation

Combinatorial optimisation (Papadimitriou & Steiglitz, 1998) involves finding the optimal solution(s) from in a discrete solution search space. A combinatorial optimisation problem comprises one or more of the subsequent three tasks, i.e. subset selection, partitioning, and permutation (Venkatesh & Singh, 2018).

A subset selection task involves picking a subset of items or objects from a given set. An example of subset selection based combinatorial optimisation problems is the Knapsack Problem (KP). When solving a KP, one is given a set of items, each with a weight and a price, as well as a knapsack with a maximum limit of weight capacity. The task is to collect a subset of these items which their weights do not exceed the knapsack capacity, while maximising the total price of the collected items. A number of KP variants have been posed in the literature includes 0-1 KP (Cao et al., 2018), multi-dimensional KP (Chih, 2018), dynamic KP (Kleywegt & Papastavrou, 1998), and multi-objective KP (Mansour et al., 2018).

Partitioning is also known as grouping or clustering. It involves separating a given set of items or objects into various partitions. Examples of partitioning based combinatorial optimisation problems include data clustering (Amiri & Dehkordi, 2018), Image clustering (Rahkar Farshi et al., 2018), and protein sequence clustering (Steinegger & Söding, 2018).

A permutation based combinatorial optimisation problem involves arranging a given set of objects in certain order. For example, the TSP is a permutation based combinatorial optimisation problem which aims to identify the shortest Hamiltonian path of a given sets of cities. Besides TSP, other examples of permutation based combinatorial optimisation problems include PFSP (Framinan et al., 2004, Makrymanolakis et al., 2016), and Quadratic Assignment Problem (QAP) (Dokeroglu et al., 2019).

There are combinatorial optimisation problems that involve more than one of the three tasks (i.e. subset selection, partitioning, and permutation), such as the TTP (Bonyadi et al., 2013) and the *k*-Interconnected Multi-Depot Multi TSP (*k*-IMDMTSP) (Venkatesh & Singh, 2018). Specifically, TTP concurrently requires subset selection and permutation, while *k*-IMDMTSP involves all the three tasks. Most combinatorial optimisation problems are NP-hard. Therefore, robust methods are essential to effectively solve such problems.

2.2 Approximation Algorithms

As mentioned in Chapter 1, there are two main categories of approaches for solving a combinatorial optimisation problem, i.e. exact and approximation algorithms. Exact algorithms employ deterministic mathematical models to obtain the global optimal of a combinatorial optimisation problem, while approximation algorithms utilise heuristics and iteration improvement to explore the solution search space. This sub-section provides a review on various existing approximation algorithms.

Approximation algorithms can be categorised into two sub-classes, i.e. single-point based models and multi-point based models. In a single-point based model, only a single solution is maintained in the memory. In contrast, a multi-point based model maintains a population of solutions in the memory. The following sub-sections review the two sub-classes of models.

2.2.1 Single-point based Models

A single-point based model is also known as a trajectory based model. In a single-point based model, only a single solution is maintained in the memory. In each iteration, a new solution is produced by modifying the current solution. If the newly produced solution is accepted with certain criteria, it replaces the current solution and will be used in the following iterations. Examples of single-point based model include hill-climbing, SA (Van Laarhoven & Aarts, 1987) and ILS (Lourenço et al., 2010).

Hill-climbing (Selman & Gomes, 2006) is a single-point based model which utilises Only Improvement as its acceptance criteria. In each iteration, a new solution is produced by perturbing the current solution with a perturbative LLH. If the newly produced solution is accepted to replace the current solution if not only if it has a better quality than the current solution. A drawback of this hill-climbing model is that it can be easily trapped in a local optimal solution.

SA is inspired by the annealing process of metals (Van Laarhoven & Aarts, 1987). In each iteration of SA, a new solution is produced by transforming the current solution using a perturbative LLH. If the newly produced solution has a better quality than the current solution, it is accepted. Otherwise, a deteriorated solution is accepted based on a probability that is proportional to the current temperature. The idea of the cooling schedule in the annealing process is computationally realised as a step-by-step decrement of the probability to accept a worse solution along the optimisation process. The mechanism of SA is employed as a move acceptance method in single-point based hyper-heuristics, as mentioned in Section 2.3.2.

ILS is an approximation algorithm which consists of two phases, i.e. a diversification phase and an intensification phase (Lourenço et al., 2010). In ILS, the incumbent solution iteratively goes through this two phases before an acceptance decision is made. During the diversification phase, a new solution is proposed by performing a perturbation to the incumbent solution. After the diversification phase, the intensification phase is initiated to perform local search based on the newly proposed solution.

2.2.2 Multi-point based Models

As compared with a single-point based model, a multi-point based model maintains a population of solutions in the memory. Therefore, a multi-point based model is also known as a population based model. The two main branches of population based models are evolutionary computation and swarm intelligence (Pandi & Panigrahi, 2011). Evolutionary computation models are inspired by principles of biological evolution such as recombination (crossover), mutation, and selection. Examples of evolutionary computation methods include GA, Evolutionary Strategy, and Evolutionary Programming. Swarm intelligence algorithms are based on the emergent collective behaviours of a population of interacting individuals in adapting to the local and/or global environment. Examples of swarm intelligence algorithms include Particle Swarm Optimisation (PSO) (Kennedy, 2010), Ant Colony Optimisation (ACO) (Dorigo, 1992, Dorigo et al., 2006), and Bee-inspired models (Lučić, 2002, Karaboga, 2005, Pham et al., 2006, Marinakis et al., 2011, Wong, 2012).

Bee-inspired models are a popular class of swarm intelligence models inspired by the collective bahaviours of bees. Bees are highly organised social insects. Their survival relies on assigning an important task to each bee in a cooperative mode. The tasks include mating, reproduction, foraging, and constructing hive. These behaviours can be computationally realised as algorithmic tools to solve various optimisation problems. Examples of bee-inspired models include the Honey Bees Mating Optimisation (HBMO) model, the Bee System (BS), the Bees Algorithm (BA), the Bee Colony Optimisation (BCO) model, the OptBees model and the Artificial Bee Colony (ABC) model. Table 2.1 lists the related studies on these models, as follows.

Table 2.1 Related works of bee-inspired models

Bee-inspired models	**Related Works**
Honey Bees Mating Optimisation (HBMO)	Abbass (2001) Afshar et al. (2007) Jahanshahi and Haddad (2008) Fathian et al. (2007) Marinakis et al. (2011)
Bee System (BS)	Lučić (2002) Davidović et al. (2012) Girsang et al. (2012) Nikolić and Teodorović (2013) Stojanović et al. (2015)
Bees Algorithm (BA)	Pham et al. (2006) Ang et al. (2009) Sagheer et al. (2012) Karaarslan (2013)
Bee Colony Optimisation (BCO)	Wong et al. (2008a) Wong et al. (2008b) Wong et al. (2010a) Wong et al. (2010b) Wong (2012) Wun et al. (2014) Wong and Choong (2015) Choo et al. (2016)
OptBees	Maia et al. (2012) Maia et al. (2013) Cruz et al. (2013a) Cruz et al. (2013b) Cruz et al. (2016) Masutti and De Castro (2016)
Artificial Bee Colony (ABC)	Karaboga (2005) Banharnsakun et al. (2010) Karaboga and Gorkemli (2011) Li et al. (2011) Akay et al. (2011) Tasgetiren et al. (2011) Karabulut and Tasgetiren (2012) Li and Yin (2012) Kıran et al. (2013) Venkatesh and Singh (2015) Zhong et al. (2017) Venkatesh and Singh (2018)

Abbass (2001) proposed a Honey Bees Mating Optimisation (HBMO) model. The HBMO model is inspired by the mating behaviour of honey bees. HBMO consists of three components, which includes an initialisation procedure, a number of mating flights, and a breeding phase. During the mating flights, crossover heuristics are used for the mating flights to combine the solutions represented by the queen and a number of drones. After that, the queen start breeding and workers are assigned to enhance the