A Study of Pheromone Maps for Ant Colony Optimization Hyper-Heuristics

By
Emilio Singh

Submitted in fulfilment of the requirements for the degree of Philosophiae Doctor in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria
January 2022

ABSTRACT

In recent years, there has been an increasing development of hyper-heuristics in the field of combinatorial optimisation. Broadly speaking, the term hyper-heuristic refers to a technique or algorithm that aims to provide a more generalised solution to, usually, a combinatorial problem. Hyper-heuristics differ from other combinatorial solution methods by working primarily in the heuristic space, as opposed to the solution space, to create more generalisable solutions for problems. There are four types of hyper-heuristics: generation constructive (GC), generation perturbative (GP), selection constructive (SC) and selection perturbative (SP). Each type functions by either generating new heuristics or by selecting which existing heuristics to apply to a problem. They are further delineated by whether the hyper-heuristic is constructive or perturbative with the former making solutions from scratch and the latter modifying and refining existing solutions.

Despite increasing research into hyper-heuristics, one area where research has been lacking is in the use of ant algorithms by hyper-heuristics to drive the search through the heuristic space. While there have been some investigations into the employment of ant algorithms by hyper-heuristics, a comprehensive study into the use of ant algorithms and in particular their central search mechanism, the pheromone map, has largely not been done. This research endeavours to investigate and study the use of ant algorithms by the four different types of hyper-heuristic to search the heuristic space. The goal is to improve the employment of ant algorithms by hyper-heuristics through the study of how the pheromone map can be used to explore the heuristic space.

A general ant algorithm for searching the heuristic space (HACO) was presented and extended for each of the four types of hyper-heuristics. This investigation specifically focused on examining the impact that using different

pheromone maps (1D, 2D and 3D) would have on the ant algorithms used by the hyper-heuristics. Furthermore, a hybrid algorithm (HACOH), one that combines multiple HACO algorithms with their pheromone maps, was presented to improve upon the use of the different pheromone maps.

The proposed algorithms (HACO and HACOH) were evaluated in multiple problem domains based on their use as one of the four hyper-heuristics. The SC and SP experiments were performed in the quadratic assignment problem (QAP) and movie scene scheduling problem (MSSP) domains. The GC experiments were conducted in the one-dimensional bin packing problem (1BPP) and MSSP domains and finally, the GP experiments were conducted in the capacitated vehicle routing problem (CVRP) and MSSP domains. These algorithms were assessed primarily in terms of optimality and generality although consideration of runtimes and comparisons with existing heuristics was included as well.

The results showed that there were statistically significant differences between the different pheromone maps when used in ant-based hyper-heuristics across a wide number of the problem domains. The only exception was the SC-MSSP experiments where differences were observed but not significant. In these experiments, at least one type of pheromone map emerged as suboptimal for use in the hyper-heuristic in the problem domain. It was not always the case that a single type of pheromone map would predominate over the others, but the results indicated clear delineations between better or worse pheromone maps to use in hyper-heuristics across the domain experiments. The HACOH algorithm showed some promise in use in the generation hyper-heuristics, in the 1BPP and MSSP domains, but was generally inferior to a non-hybrid HACO algorithm in the majority of the experiments, indicating that the hybrid algorithm is not universally superior to its non-hybrid counterparts.

These results have met the research objectives of this thesis by showing that, firstly, ant algorithms can be employed successfully, by all four types of hyper-heuristics. More importantly, however, the results showed that there are meaningful differences between the use of the different pheromone maps in ant-based hyper-heuristics and that choosing the optimal map for an ant-based hyper-heuristic depends on the problem domain among other factors.

Keywords: ant-colony optimisation, hyper-heuristics, discrete optimisation

Supervisor : Prof. Nelishia Pillay

Department : Department of Computer Science

Degree : Doctorate in Computer Science

Declaration: Plagiarism

I, Emilio Singh, declare that:

i the research reported in this thesis, except where otherwise indicated or acknowledged, is my original work;

ii this thesis has not been submitted in full or in part for any degree or examination to any other university;

iii this thesis does not contain other persons data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons;

iv this thesis does not contain other persons writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:

(a) their words have been re-written but the general information attributed to them has been referenced;

(b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced;

(c) this thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the References sections.

Signature: .....

Emilio Singh May 12, 2022

Declaration: Supervisor

I confirm that this work was done under my supervision and it is the candidate's original work. As the candidate's supervisor, I have approved this thesis for submission.

Signature: .....

Nelishia Pillay May 12, 2022

Declaration: Publications

The following publications are associated with the research presented in this thesis:

1. E. Singh and N. Pillay, "Ant-Based Generation Constructive Hyper-Heuristics for the Movie Scene Scheduling Problem," Theory and Practice of Natural Computing 109–120, 2021
2. E. Singh and N. Pillay, "Ant-Based Hyper-Heuristics for the Movie Scene Scheduling Problem," Artificial Intelligence and Soft Computing 342–353, 2021
3. E. Singh and N. Pillay, "A Study of Ant-Based Pheromone Spaces for Generation Constructive Hyper-Heuristics," Swarm and Evolutionary Computation, Accepted Before 2022
4. E. Singh and N. Pillay, "A Study of Ant-Based Pheromone Spaces for Selection Constructive Hyper-Heuristics," Annals of Operations Research, under review, 2020
5. E.Singh and N. Pillay,"A Study of Transfer Learning in an Ant-Based Generation Construction Hyper-Heuristic", IEEE CEC, Accepted Before 2022

Acknowledgements

I would like to sincerely thank my supervisor, Professor Nelishia Pillay. She introduced me to the field of hyper-heuristics and expertly, and patiently, guided me in completing this research. Without her supervision, this would have been far more difficult to complete. I am also thankful for the services of the MIT cluster, without which the experiments in this research would have not been possible.

I would also like to thank my family for their endless support for my academic adventures and their continuous belief in my abilities. Anil, Pamela and Sergio, I will love you forever. Lastly, I would like to acknowledge my late grandmother, Kay, whose earnest love for me and belief in my abilities inspired me to pursue education. I wish she could have been here to see this completed, but this will have to suffice.

List of Abbreviations

ACO	Ant Colony Optimisation
GP	Genetic Programming
GA	Genetic Algorithm
CVRP	Capacitated Vehicle Routing Problem
1BPP	One-Dimensional Bin Packing Problem
2BPP	Two-Dimensional Bin Packing Problem
QAP	Quadratic Assignment Problem
MSSP	Movie Scene Scheduling Problem
SACO	Simple Ant Colony Optimisation
AS	Ant System
FANT	Fast Ant System
TS	Tabu Search
VNS	Variable-Neighbour Search
EA	Evolutionary Algorithms
HACO	Hyper-heuristic Ant Colony Optimisation
HACOH	Hyper-heuristic Ant Colony Optimisation Hybrid
SC	Selection Constructive
SP	Selection Perturbative
GC	Generation Constructive
GP	Generation Perturbative
SDD	Standard Deviation of Distances

Table of Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

1.1 Purpose of the Study

The field of hyper-heuristics is one with the aim of creating generalisable solutions to combinatorial problems, and it achieves this through working in a heuristic space that is distinct from the solution space [1]. Research into hyper-heuristics has resulted in the development of four main types which are the selection constructive, selection perturbative, generation constructive and generation perturbative hyper-heuristics [2]. Selection hyper-heuristics revolve around choosing and applying existing low-level heuristics whereas construction hyper-heuristics revolve around creating new heuristics. Constructive and perturbative in this context refer to whether the underlying solution will be created from scratch or modified from an existing prior form respectively. Subsequent chapters will define these concepts in greater detail.

However, despite research into these types of hyper-heuristics, the existing work has largely fixated on a few well-known techniques, like genetic algorithm (GA), genetic programming (GP), and other evolutionary algorithms. Beyond some limited applications in selection constructive hyper-heuristics [3], ant algorithms have not been widely studied for selection perturbative hyper-heuristics and generation hyper-heuristics to search the heuristic space. Moreover, the existing research has purely focused on limited applications of ant algorithms without consideration of how the algorithms could be best applied by hyper-heuristics to search the heuristic space. This has left open potential for the development of ant-based hyper-heuristics for generation hyper-heuristics. More importantly, this demonstrates that there is a void in terms of the research regarding how the ant-based algorithms are used as hyper-heuristics themselves.

Specifically, in the past, selection hyper-heuristics have employed ant algorithms primarily fusing the ant algorithm to search the heuristic space in the same way that ant algorithms are applied to search the solution space. This research hypothesises that this is a sub-optimal method and that a better study of how ant algorithms could be employed to search the heuristic space would improve ant-based hyper-heuristics.

Hence the purpose of this research is to do a comprehensive investigation into improving how the operating mechanism of ant algorithms, the pheromone map, could be better studied for improving the use of ant-based hyper-heuristics. A central hypothesis of this thesis is that different types of pheromone maps will greatly impact the ability of the ant-based hyper-heuristic to succeed. Specifi-

cally, this investigation aims to examine the impact of different types of pheromone maps (1D, 2D, and 3D) in how they influence the performance of ant-based hyper-heuristics. Further justification is provided in Section 2.6.

1.2 Objectives

The primary goal of this study is to investigate the effects that different pheromone (1D, 2D, and 3D) maps have on ant-algorithms used in the four types of hyper-heuristics. To achieve this goal several objectives have to be met.

1. To design and implement an ant-based selection constructive hyper-heuristic with 1D, 2D, and 3D pheromone maps.
2. To design and implement an ant-based selection perturbative hyper-heuristic with 1D, 2D, and 3D pheromone maps.
3. To design and implement an ant-based generation constructive hyper-heuristic with 1D, 2D, and 3D pheromone maps.
4. To design and implement an ant-based generation perturbative hyper-heuristic with 1D, 2D, and 3D pheromone maps.
5. To design and implement a method such that multiple ant algorithms, each using a different pheromone map, can be hybridised together.
6. To assess the effect of 1D, 2D and 3D pheromone maps have on the four types of ant-based hyper-heuristics as well as to compare the hybrid algorithm against the non-hybrid algorithms

1.3 Thesis Scope

The scope of this thesis is the use of ant colony optimisation (ACO) algorithms as the basis for four types of hyper-heuristics and the evaluation of the effect that different pheromone maps have on these hyper-heuristics. The scope is as follows:

- Ant Algorithms: There are two types of ant algorithm that are considered, and adapted where necessary, as the operating mechanism for the various hyper-heuristics in this thesis. They are the ant system algorithm [4] and the fast ant algorithm [5].
- Hyper-Heuristics: The scope of this thesis is restricted to the four types of four hyper-heuristics. They are selection constructive, selection perturbative, generation constructive, and generation perturbative hyper-heuristics.
- Optimisation Problems: This thesis considers 4 discrete combinatorial optimisation problem domains. All of these problems are minimisation problems that have a single objective value that is to be minimised.
- Problem Domains: There are four domains considered in this thesis. These are the capacitated vehicle routing problem (CVRP), the one-dimensional bin packing problem(1BPP)[6], the quadratic assignment problem (QAP), and the movie scene scheduling problem (MSSP). The first three domains are

common domains that have been widely used in hyper-heuristic and discrete combinatorial optimisation research. The last domain was chosen because it is a more recent domain in the field and has not been used before. The choice, therefore, of domains reflects a desire to use old and new domains to study the different types of ant-based hyper-heuristics in this research. Broad coverage of different types of optimisation problems also provides a good platform for assessing the application of the ant algorithms to hyper-heuristics.

The focus of this thesis is on improving the application of ant algorithms in hyper-heuristics specifically. It is not in competing or replacing existing hyper-heuristic methods.

1.4 Contributions

The main contribution of this thesis is the comprehensive study of how different types of pheromone maps can influence the operating capacity of various ant-based hyper-heuristics. To the knowledge of the author, this is the first study of its kind that provides an in-depth analysis of how hyper-heuristics can employ ant algorithms to search the heuristic space, including hyper-heuristics that have not employed any form of ant algorithm in the past.

1. A comprehensive study of the effects of different types of pheromone maps in terms of how those affect ant-based hyper-heuristics across all four types of hyper-heuristics.
2. The employment of ant algorithms by a generation constructive and generation perturbative hyper-heuristic is a truly novel application that has demonstrated that ant-based methods can be utilised far outside their original operating scope.
3. The hybridisation of the different ant-based hyper-heuristics, with distinct pheromone maps, has demonstrated that the effects of using different pheromone maps, with their relative advantages and disadvantages, can be combined into a single algorithm that offers comparable performance to any single pheromone map without requiring the different maps to be selected based on the problem.

1.5 Thesis Layout

The rest of the thesis is organised as follows:

1.5.1 Chapter 2 - Background

This chapter provides the necessary background information pertinent to this thesis and surveys the related work. The chapter also provides information about discrete optimisation problems, ant algorithms, and their existing employment

by hyper-heuristics and a discussion of the four kinds of hyper-heuristics. Finally, background information for the problem domains used in this research is presented here.

1.5.2 Chapter 3 - Research Methodology

Chapter 3 presents a brief description of existing research methods in Computer Science. It also provides an explanation of how the appropriate research methodology is applied to meet the objectives indicated in Section 1.2. A detailed description of the problem domains and the instances is presented as well as a description of the experiments done for the thesis and the necessary parameter values and assessment metrics.

1.5.3 Chapter 4 - Pheromone Maps

In this chapter, a detailed description of the central operating mechanism of the ant algorithm, the pheromone map, is given. The focus of this chapter is specifically on how the pheromone map will be modified and adapted for this research. Namely with the methods of compression and projection that are used in producing different kinds of pheromone maps outside of the native two-dimensional form.

1.5.4 Chapter 5 - Hyper-Heuristic Ant Colony Optimisation

This chapter details the basics of how the ant algorithms are used by hyper-heuristics. This covers a generalised form of the ant-based hyper-heuristic that broadly applies to all types of hyper-heuristics. Details specific to each hyper-heuristic will be provided in their chapters.

1.5.5 Chapter 6 - Ant-Based Selection Hyper-Heuristics

In this chapter, a detailed description of how an ant algorithm is employed by the two types of selection hyper-heuristics is provided. This covers selection constructive and selection perturbative hyper-heuristics. The reason is that selection hyper-heuristics do not differ much for both constructive and perturbative tasks. This chapter includes the details of the application of hyper-heuristics to the MSSP and QAP domains. Finally, parameters that are specific to the hyper-heuristics are presented here.

1.5.6 Chapter 7 - Ant-Based Generation Constructive Hyper-Heuristic

In this chapter, a detailed description of an ant-based generation constructive hyper-heuristic is provided. This includes the details of the application of the hyper-heuristic to the 1BPP and MSSP domains. Finally, parameters that are specific to this hyper-heuristic are presented here.

1.5.7 Chapter 8 - Ant-Based Generation Perturbative Hyper-Heuristic

This chapter provides a detailed description of the ant-based generation perturbative hyper-heuristic. This includes the details of the application of the hyper-heuristic to the CVRP and MSSP domains. Finally, parameters that are specific to this hyper-heuristic are presented here.

1.5.8 Chapter 9 - Hybridising Ant-Based Hyper-Heuristics

In this chapter a detailed methodology for an algorithm that hybridises the algorithms that use the three types of pheromone maps is presented. This covers the hybridisation scheme and its operating parameters and conditions.

1.5.9 Chapter 10 - Results and Discussion

This chapter provides the results of the experiments in this thesis. The results are assessed in terms of their optimality and generality in their respective domains with comparisons to existing heuristics and techniques provided where necessary. This chapter also includes an analysis of the pheromone maps with the appropriate graphical representation.

1.5.10 Chapter 11 - Conclusion and Future Work

Finally, this chapter provides the conclusion of this thesis and presents ideas for future work.

CHAPTER 2

Background

2.1 Introduction

This chapter provides the foundation for the research presented in this thesis. This includes both a background on topics relevant to this research as well as a justification for it. The rest of this chapter is organised as follows. Section 2.2 provides a brief overview of discrete combinatorial optimisation problems as these are the problems considered in this research. Section 2.3 provides an explanation of ACO algorithms upon which the basis of this research is conducted. Section 2.4 introduces the concept of hyper-heuristics and details the four types. Section 2.5 provides a summary of the existing applications of ACO to hyper-heuristics. Section 2.6 provides the critical analysis of relevant literature and finally, Section 2.8 provides a summary for the entire chapter.

2.2 Discrete Combinatorial Optimisation

Combinatorial optimisation is a type of optimisation task that concerns itself with finding an optimal object or arrangement of objects from a finite set of them [7,8]. The use of the phrase discrete refers to the fact that these problems revolve around indivisible objects that do not operate in the real-valued domain.

A formal definition, adopted from [9], for a combinatorial optimisation problem is as follows:

- A set of variables $X_k = \{x_1, x_2, ..., x_n\}$ which defines all possible values of x_k.
- A set of constraints, C that impose relationships and requirements on the variables.
- An objective function, f be optimised.

In general, there are two types of combinatorial optimisation problems: maximisation and minimisation problems. This research concerns itself with minimisation problems but in principle, every maximisation problem can be converted to a minimisation problem by negating f.

An important part of the problem is feasibility. Feasibility in this context refers to a solution to a combinatorial optimisation problem that does not violate any of the problem's constraints. Solutions that exist that violate any of the constraints are called infeasible solutions. The goal of any good technique is to

move away from the infeasible regions of the search space, to the feasible regions with good solutions. Therefore, the set of all feasible solutions, also called the search/solution space S, can be defined as follows:

$$S = \{s = \{(x_1, v_1), (x_2, v_2), ..., (x_n, v_n)\} |\ v_i \in D_i\} \tag{2.1}$$

where v_i refers to a specific value taken from D_i, a set of possible values, and assigned to the variable x_i and s satisfies all constraints.

2.3 Ant-Colony Optimisation

The term ant algorithm broadly refers to a category of algorithms that are modelled on particular behaviours of organic ants in real life [10]. These behaviours, for example, foraging behaviour, have been developed in an algorithm by Dorigo in [11]. Since then, the field of research has grown into a suite of various kinds of ant-based algorithms that rely on the same fundamental behaviours described in the original algorithm to solve a variety of problems, most notably combinatorial and search problems [11,12].

The basis of the functioning of most ant algorithms is derived from the communication of information between the ants and their search process for a given solution to a problem. This is an application of stigmergy [12] whereby the ants use an artificial replication of natural ant pheromones to communicate information about their searches. As the group of ants engage in searches, a tendency towards the increase of pheromone in good areas of the search space gradually produces better and better solutions. Like any stochastic, evolutionary-based optimisation method, there is no hard guarantee that the found solution will always be the best, especially in high dimensionality problems but unlike many, it does come with a guarantee of convergence although the speed of convergence is problem dependent [13].

Nevertheless, ant algorithms are an attractive choice for combinatorial problems for many reasons. Their ability to function on any problem that has a graph-based representation makes them applicable to a wide domain of problems [14]. The shared information of the ants also leads to rapidly improving solutions as well as the ability to more readily adapt to dynamic or changing problem environments [14].

2.3.1 Ant Algorithm Operation

To better illustrate the operation of ACO, consider the following simple graph of nodes in Figure 2.1 and the problem of trying to find the lowest cost route between Node A and Node G. This problem is purely for illustrative purposes. Ant algorithms can be applied to any problem with a graph-based representation.

This graph, $G = (V, E)$ where V is a set of nodes and E is a matrix representing the costs to move between nodes (links i to j), is a common representation for a common optimisation problem, namely the shortest path problem [15]. In

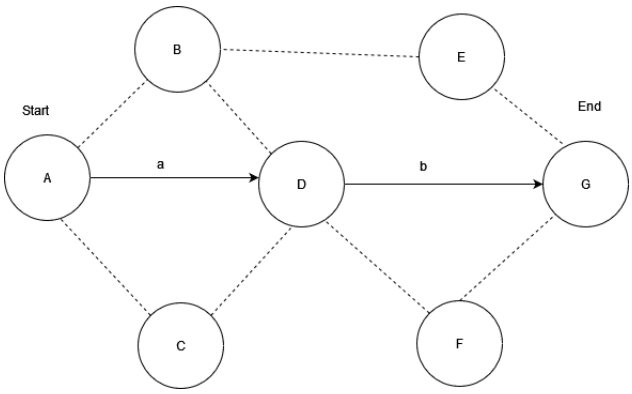

Fig. 2.1: Graph for Shortest Path Problem

this graph, two pheromone concentrations are given as a and b on the links of the graph. For the purposes of this problem, the graph can be considered to be directed although this is less relevant to the principle at hand. In the graph, a possible solution is indicated with solid arrows. Namely, the path of $A \to D \to G$. The dotted lines represent other possible connections.

To illustrate a generalised functioning of an ant algorithm, consider the Simple Ant Colony Optimisation algorithm (SACO) [16]. This is one of the simplest versions of the ACO and serves to illustrate the generalised functioning of an ACO algorithm without unnecessary, at this point, details.

For SACO, the algorithm starts with all edges initialised with small random values for the initial pheromone values. An ant, the decision-making agent, is placed at the start node and then decides which edges to move to. This can be generalised to k ants but is kept to a single ant for simplicity.

During each iteration of the SACO, the ants will construct paths from the start node to the end node using a decision policy to decide which of the nodes to move to.

If ant k is at node i, then it will select the next node $j \in N_i^k$ based on the probability rule [16],

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{j \in N_i^k(t)} \tau_{ij}^\alpha(t)} & \text{if } j \in N_i^k(t) \\ 0 & \text{if } j \notin N_i^k(t) \end{cases} \tag{2.2}$$

where N_i^k is the set of feasible nodes connected to node i and ant k's position in the graph. The variable t denotes the current iteration of the algorithm. For the sake of simplicity, ignore the potential for loops.

In Equation 2.2 α is a positive constant that magnifies the influence of the pheromone concentration at the indices of (i, j) in the pheromone map. Larger values increase the significance afforded to larger concentrations of pheromone and more heavily influence the ant's search direction.

Once the ant has traced its path from the start to the end node, it deposits some amount of pheromone on each link visited, in the reverse order of traversal. The ant drops a pheromone amount,

$$\tau_{ij}^{k}(t) \propto \frac{1}{L^{k}(t)} \tag{2.3}$$

where $L^k(t)$ is the length of the path constructed by ant k at iteration t. The notation choice here refers to the pheromone, τ_{ij}^{k}, deposited by ant k specifically. This pheromone is deposited in a two-dimensional matrix called a pheromone map, which is structured identically to the edge matrix. It serves as a collective knowledge store of all the ants and their efforts to solve the problem. Over several ants, the pheromone for a given link at t is merely the sum of all ants' deposited pheromone on that link. For this example, the fitness of the path is its length although this can vary based on the problem. Different measures can be used to decide how much pheromone should be deposited and this is discussed in Sections 2.3.2 and 2.3.3.

The last component of a generalised ant algorithm is the management of pheromone over time. An initial problem with SACO was the premature convergence of the ants on paths that became saturated with pheromone early on [17]. To prevent this, pheromone evaporates, on a link (i, j) at a rate given by,

$$\tau_{ij}(t) = (1 - p)\tau_{ij}(t) \tag{2.4}$$

where $p \in [0, 1]$. The notation in this context refers to the pheromone, τ_{ij}, in the map at indices (i, j) for evaporation which does not require reference to any ant k. The constant p controls the rate of evaporation which enables some information contained on a link to evaporate and be forgotten, allowing other links to be explored. Taken together, the SACO algorithm is formalised into Algorithm 1.

Algorithm 1: SACO Path Construction Process

```
1  initialise τ_ij(0) to small random values;
2  t = 0;
3  Place n_k ants on the origin node;
4  while t < t_max do
5      foreach ant k = 1, ..., n_k do
6          Construct a path x^k(t);
7          x^k(t) = ∅;
8          while path incomplete do
9              Select the next node using Equation 2.2;
10             Add link (i, j) to path x^k(t);
11         Calculate path length of f(x^k(t));
12     foreach link (i, j) do
13         Evaporate pheromone using Equation 2.4;
14     foreach ant k do
15         foreach link (i, j) of x^k(t) do
16             δτ^k = 1 / f(x^k(t));
17             Update τ_ij(t);
18     t = t + 1;
19 Return the best path found;
```

This algorithm describes the most basic operation of an ant algorithm. In Line 4, the terminating condition is an iteration limit specified by t_{max} but other termination criteria can be used as well. It is a cooperative process between all of the ants each of which contributes some amount of information about the problem they are trying to solve with the others through the shared access and maintenance of, pheromone accumulation.

2.3.2 Ant System

Dorigo [11] developed the first ant algorithm called the ant system (AS). While the SACO is meant to demonstrate the basic principles of ant algorithms, the AS is the more recognisable form of the canonical algorithm [18].

The AS retains most of the features of operation that are recognisable in the SACO except for two small, but significant, changes. The first is the use of a different probability transition rule. This is given as,

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)\eta_{ij}^\beta(t)}{\sum_{j \in N_i^k(t)} \tau_{ij}^\alpha(t)\eta_{ij}^\beta(t)}, & \text{if } j \in N_i^k(t) \\ 0, & \text{if } j \notin N_i^k(t) \end{cases} \tag{2.5}$$

In this new probability transition rule, η_{ij} represents the attractiveness of moving to a given node based on some external heuristic. In this way, the

pheromone associated with moving to a node, τ_{ij} is now weighed against a heuristic score of moving to that node, with α and β serving as control coefficients that determine the magnitude of the respective influences of both components.

The other change to the formulation is the use of a tabu list to prevent loops from being formed. This is a necessary inclusion to remedy one of the issues of the native SACO implementation.

2.3.3 Fast Ant System

The fast ant system (FANT) was developed by Taillard and Gambardella [5,19] as a specific solver for the QAP. As an ant algorithm, it has some major deviances from the more standard formulation. The first deviation is the use of only a single ant, hence the name. Other ant algorithms typically rely on the cooperation and interaction of multiple ants, as agents, to facilitate the search process. The FANT algorithm makes use of a single ant to greatly reduce its computational complexity.

The second significant departure from conventional ant algorithms is the probability transition rule. The rule is given as,

$$p_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)}{\sum_{j \in N_i^k(t)} \tau_{ij}^{\alpha}(t)}, & \text{if } j \in N_i^k(t) \\ 0, & \text{if } j \notin N_i^k(t) \end{cases} \tag{2.6}$$

This is very similar to the AS update rule with the noted omission of η. This is because $\beta = 0$ in the FANT system as it deliberately omits heuristic information in favour of a pure reliance on pheromone information to guide the search.

The third and final deviation is in the pheromone update rule. This is given as,

$$\tau_{ij}(t+1) = \tau_{ij}(t) + w_1 \Delta\tilde{\tau}_{ij}(t) + w_2 \hat{\tau}_{ij}^{+} \tag{2.7}$$

where w_1 and w_2 are control parameters that weight the influence of the current solution, $\tilde{\tau}_{ij}$ at t and the best solution, $\hat{\tau}_{ij}^{+}(t)$, found so far. A link receives pheromone depending on whether it is included in the current solution or the best solution. Therefore, the pheromone values are calculated as,

$$\Delta\tilde{\tau}_{ij}(t) = \begin{cases} 1 & \text{if } (i,j) \in \tilde{x}(t) \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

and

$$\Delta\hat{\tau}_{ij}(t) = \begin{cases} 1 & \text{if } (i,j) \in \hat{x}(t) \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

where $\tilde{x}$ and $\hat{x}$ are the current path and best path respectively.

In this way, the reinforcement provided to the pheromone is relatively large and purely based on occurrence in paths, regardless of the path's actual length.

These modifications require further changes to the underlying algorithm. In particular, all of the pheromones must be initialised to 1 at execution and whenever a new best solution is found, the pheromone must be reinitialised again. This allows the algorithm to exploit the area of the new best path. Finally, if the current solution is the same as the best, the value of w_1 is increased. As w_1 controls the influence of the current path, this diminishes the influence of the best path to encourage additional exploration.

2.4 Hyper-Heuristics

The term hyper-heuristics refers to a field of research that aims to provide generalisable solutions to combinatorial optimisation problems by working in the heuristic space instead of the solution space [2]. The solution space refers, broadly, to the range of possible direct solutions for a given problem whereas the heuristic space refers to a range of possible choices relating to heuristics that can then be applied to solve the problem [1,20]. This works at a level of abstraction one step removed from the direct solution. Broadly speaking, hyper-heuristics are either used to select from an existing pool of low-level heuristics or use some technique to generate new ones.

A low-level heuristic is simply any kind of heuristic procedure or method that can be applied to solve a problem and can be either constructive or perturbative [2]. These methods are meant to solve a wide range of instances within a problem domain. A constructive heuristic builds a solution to a problem from scratch whereas a perturbative heuristic modifies or perturbs an existing solution in the hope of refining its quality.

Hence, four kinds of hyper-heuristics relate to how the hyper-heuristic interacts with low-level heuristics and what kind of low-level heuristics that interaction is meant to facilitate [21,22]. These are selection constructive, selection perturbative, generation constructive and generation perturbative. These types are discussed in Sections 2.4.1–2.4.4.

2.4.1 Selection Constructive Hyper-Heuristics

A selection constructive hyper-heuristic is a hyper-heuristic that continuously selects a low-level heuristic to apply to a problem in a particular state until a solution for that problem is fully constructed [2].

That is, given a set of low-level construction heuristics H and a problem P, a selection constructive hyper-heuristic will gradually construct a solution s for P by choosing and applying heuristics from H until the problem is solved. A generalised form of this algorithm is given in Algorithm 2. This algorithm represents a generalised version of a selection constructive hyper-heuristic, and as such, makes use of a termination criterion that applies to all of them. Namely, stopping once a complete solution is made. It is possible to use different criteria as termination conditions but that would be more problem or implementation-specific, such as accounting for partial completion of the solution.

Algorithm 2: Generalised Selection Constructive Hyper-Heuristic

Data: problem p, a set of construction heuristics H
Result: a solution s

```
1 s = ∅;
2 while s is incomplete do
3 |   Select heuristic h from H;
4 |   Apply h to s;
```

There are several different methods of choosing the heuristic from the set. The first type is called case-based reasoning. With case-based reasoning, the idea is to solve new problems based on previously solved old problems [23]. That is, to create a set of solved problems or cases, and use this to solve new instances based on their similarity to any of the prior solved cases. This is a simple solution to the problem but comes with the challenges of building an effective case base as well as choosing the appropriate similarity metric for the comparison of cases. In practice, case-based reasoning has been successfully employed by several selection constructive hyper-heuristics in many domains like examination timetabling [24,25].

Another option is to use a local search method to search the heuristic space [2]. Local search methods like tabu search (TS) [26] and variable-neighbour search (VNS) [27] can be used to search the heuristic space in the way that they are used to search a solution space. In these cases, the algorithms explore the heuristic space of a heuristic combination that is a set of heuristics where a given heuristic h can be applied. Every heuristic can be applied at least once with the total size of the heuristic combination, and thus the number of heuristics, dependent on the problem. An example of a heuristic combination is $h_0h_1h_2h_3$.

In contrast to the local search methods, which only explore a single point in the heuristic space at a time, population-based methods are capable of searching multiple points simultaneously, with evolutionary algorithms (EA) being the chief method employed to facilitate this [2]. In this case, genetic operators are used on the heuristic combinations described previously. Instead of modifying the heuristic combination one heuristic at a time, these operators use evolutionary algorithms to modify large regions of the heuristic combination which is represented as a chromosome in the EA. This technique has been applied successfully to both the examination timetabling problem [28] and the one-dimensional bin-packing problem [29].

Finally, it is possible to employ an adaptive or hybridised method to drive the hyper-heuristic. Sabar et al. [30] is one such example. They created a hybridisation between four low-level graph colouring construction heuristics for the examination timetabling problem.

2.4.2 Selection Perturbative Hyper-Heuristics

Selection perturbative hyper-heuristics are similar to selection constructive hyper-heuristics in the sense that they also select low-level heuristics. However, they select low-level perturbative heuristics to apply to an initially complete solution to improve that solution's quality [2]. The initial solution can be created with construction heuristics or through random generation, depending on the problem. Each perturbative heuristic modifies the state of the problem to improve the quality of the solution with each operation, ideally until no additional improvements can be reached.

That is, given a set of low-level perturbative heuristics H, an initial solution s and a problem P, a selection perturbative hyper-heuristic will gradually perturb the solution s for P by choosing and applying heuristics from H until no improvement is possible or another stopping condition is met.

In general, there are two types of selection perturbative hyper-heuristics: single-point and multi-point search methods. In the former case, the two most important considerations are the heuristic selection technique and the move acceptance technique [21].

Algorithm 3: Single-Point Selection Perturbative Hyper-Heuristic

```
   Data: problem p, an initial solution s_0, a set of perturbative heuristics
         H
   Result: a solution s_i
1  termination criteria = false;
2  while termination criteria == false do
3      Select heuristic h from H using the heuristic selection technique;
4      Apply h to s_i to produce s_{i+1};
5      Use move acceptance technique on s_{i+1};
6      if move accepted then
7          s_i = s_{i+1};
8      if finished == true then
9          termination criteria = true;
```

2.4.2.1 Single-Point Selection Perturbative Hyper-Heuristic A generalised form of a single-point selection perturbative hyper-heuristic algorithm is given in Algorithm 3. The algorithm can make use of a variety of different termination criteria to stop the perturbation process. A simple option is to use several iterations, but another option is to continue the process until no further improvements over a time window are observed. If the move is not accepted, denoted by Line 6, then nothing happens and the algorithm will continue. Alternative schemes extend beyond the general algorithm presented here.

There are a wide variety of different heuristic selection techniques that are applicable in a single-point selection perturbative hyper-heuristic. The most simple is the random selection strategy which just randomly chooses the next perturbative heuristic to apply [31,32,33]. On the opposite end of the spectrum is the technique which applies all of the perturbative heuristics and picks the one that ends up producing the best objective value [21]. It is also possible to make use of techniques derived from evolutionary algorithms like tournament selection and fitness proportionate selection [21].

The move acceptance technique refers to the technique used to decide if the solution produced by applying the perturbative heuristic should be retained or discarded. The most basic of all move acceptance techniques is to accept all moves [21]. This simply accepts the outcome of applying the perturbative heuristic regardless if it causes a decrease in the quality of the solution. Another technique would be to accept only improving moves [34]. This is where only moves that result in an improved solution quality are accepted. There are a large variety of move acceptance techniques and this coverage is by no means exhaustive.

2.4.2.2 Multi-Point Selection Perturbative Hyper-Heuristic Multi-point selection perturbative hyper-heuristics are so-called because they make use of a population-based method to explore the heuristic space [2]. In terms of methods, genetic algorithms [35,36,37] and particle swarm optimisation [38] have been used successfully to drive the heuristic process. In particular, genetic algorithms have been the most popular search method used by selection perturbative hyper-heuristics [2]. For their application, a multi-point selection perturbative hyper-heuristic can be used to produce a single heuristic [38] or a set of heuristics that can then be applied to an initially generated solution [35].

2.4.3 Generation Constructive Hyper-Heuristics

One of the challenges of low-level construction heuristics is deriving them, often through intensive study of the problem domains and with human input [2]. Generation constructive hyper-heuristics, therefore, aim to solve this problem by automating the process of producing constructive heuristics. The same basic goal is retained, that is to construct a solution wholly from scratch, but the key difference is that a generation constructive hyper-heuristic produces an entirely new heuristic for this process.

More specifically, given a set of problem attributes $A = \{A_0, A_1, ..., A_N\}$ and at least a problem instance i, a generation constructive hyper-heuristic will produce a new constructive heuristic h using A that can provide a solution for i. Sometimes a construction heuristic can be expanded in scope to provide solutions to previously unseen problems and this is referred to as a reusable heuristic[21]. The generation of a heuristic for a specific problem instance is referred to as a disposable heuristic. The principle approach used in generation constructive hyper-heuristics is genetic programming [39].

2.4.3.1 Representation of Low-Level Heuristics

At the most basic form, a low-level heuristic is comprised of problem-specific attributes and operators that manipulate them [2]. Two things are necessary for the success of a generation constructive hyper-heuristic. The first is that the problem attributes being used as inputs should be as basic as possible and the second is that the hyper-heuristic itself should be able to aggregate characteristics of these attributes together [40].

The source for the attributes can come from the decomposition of existing low-level constructive heuristics and problem characteristics. These attributes can then be used to create new heuristics that either represent an arithmetic function [41] or rules [42]. The former represents functions that are used to calculate priorities or desirability scores for components while assembling a solution. The latter consist of conditions and actions where the conditions enable actions to be undertaken with probabilistic branching.

2.4.4 Generation Perturbative Hyper-Heuristics

The last of the hyper-heuristic types, generation perturbative hyper-heuristics concern themselves with generating new low-level perturbative heuristics for a given problem domain or instance [2]. Like generation constructive hyper-heuristics, these create new heuristics primarily through the combination of attributes and components derived from existing heuristics. However, they apply these heuristics to fully complete solutions. Additionally, these hyper-heuristics place a greater emphasis on the use of conditional branching and iterative construct statements in their generation process, such as the if-then statement or for loops.

Generation perturbative hyper-heuristics are the least studied of all the hyper-heuristics but existing research has primarily focused around the use of genetic programming [43,44] and the grammar-based grammatical evolution (GE) [45].

One area where generation perturbative hyper-heuristics stand out is the ability of the algorithms to create new low-level perturbative heuristics that are in effect, algorithms of their own to solve problems like the travelling salesman problem [46].

The ability of the hyper-heuristic to incorporate elements like conditional-branching constructs, if-then-else statements, and iterative construct, the while loop, in conjunction with more usual operators like the logical AND, allowed the hyper-heuristic to produce heuristics that were more complicated than conventional low-level heuristics.

2.5 ACO Hyper-Heuristics

Ant algorithms have been utilised in hyper-heuristics although their application has focused solely on selection hyper-heuristics [3]. One of the challenges of studying ACO in the context of hyper-heuristics is the extremely limited research that has been done with using ACO for hyper-heuristics. In the case of generation

hyper-heuristics, there has been no prior research. However, there are still some interesting examples to draw upon.

A particular early example of an ant-based hyper-heuristic is the work of [47]. This was one of the first attempts to use ant algorithms with hyper-heuristics. The authors made use of a selection constructive hyper-heuristic driven by a modified ant system algorithm. This system was tested on the two-dimensional bin-packing problem (2BPP) and demonstrated good performance against the existing heuristics.

Another example is the application of ACO in a selective perturbative hyper-heuristic for the set covering problem [48]. The algorithm used was based on the ant system algorithm as well but modified for heuristics. The algorithm performed competitively against an ant algorithm that was optimised for the set covering the problem specifically.

In [49], the author applied an ant-based selection constructive hyper-heuristic to the travelling salesman problem. The method itself was based on the ant system algorithm and showed reasonable results in terms of being competitive with other methods like simulated annealing and GA but not being optimal in the majority of instances.

In a similar vein, in [50] the authors applied an ant-based hyper-heuristic for the travelling tournament problem, which is a related problem to the travelling salesman problem. The proposed algorithm was based on a modified version of the ant system. The results showed that the algorithm was competitive with several existing methods like simulated annealing and tabu search, although the algorithm did struggle on the larger problems. In particular, the authors noted the improved speed of the process which is an important aspect to consider for hyper-heuristics.

These examples illustrate two things. Firstly, ant algorithms have been applied successfully to create some types of hyper-heuristics but also that this development has halted around the other types of hyper-heuristics, namely generation constructive and generation perturbative. Secondly, the usage of the ant algorithm has largely crystallised into a single form that mirrors the application of ACO in non-hyper-heuristic settings. The ant algorithm is applied with some modifications to make the transition for the heuristic space but a deeper investigation of this is still needed.

2.6 Critical Analysis

Ant algorithms have been utilised in a variety of selective hyper-heuristics, primarily as the driver of the selection of heuristics for a given problem [3,49,50]. These applications of the ant algorithm to this kind of problem are natural, given that tasks like the selection of heuristics can be reformulated into graph-based searches where ants traverse a heuristic space and apply heuristics to solutions in the hope of refining them into better ones.

However, these applications have left several questions in the existing research that have stifled the potential of ACO in the field of hyper-heuristics. Firstly, the

application of ACO has fixated on the basic application of the ACO algorithm with little consideration for how the algorithm would need to be modified for the heuristic space. The second question is the lack of the use of the ACO algorithm by other types of hyper-heuristics beyond selection hyper-heuristics. Resolving these questions is the main aim of this research and the sections below will provide a greater justification.

2.6.1 Justification for Use of ACO

As discussed in this chapter, there have been some hyper-heuristics that have used ACO as the primary mechanism for driving the hyper-heuristic [3,49,50]. However, the use of ACO in hyper-heuristics has been limited to selection hyper-heuristics. One of the central aims of this research is to expand the use of ACO in hyper-heuristics by extending the use of ACO to both generation constructive and generation perturbative hyper-heuristics, which remain unexplored areas of research in the field of ACO hyper-heuristics.

Moreover, this also has the effect of advancing ant-colony optimisation research as well. One of the requirements of an ACO algorithm is that the underlying problem should be representable in a graph-based format [11]. This has had the effect of limiting where ACO algorithms could be employed as some problems are either too cumbersome to adapt to ACO or impossible. Moving the ACO into the heuristic space, across all hyper-heuristics, enables ACO and ACO-based algorithms to operate on any problem that could be solvable through hyper-heuristics, greatly expanding its potential.

2.6.2 Justification for Different Pheromone Maps

Primarily, the majority of the research into ant algorithms has focused on their utility in solving problems in the solution space. However, this has the effect of limiting ant algorithms to only problems that are structured around the basics of any given ant algorithm [11]. In particular, in the applications of ACO to hyper-heuristics performed thus far, the ACO is used in the hyper-heuristic in the same way that it would be used in the solution space. Namely, the ACO is deployed with a two-dimensional pheromone map and the task of choosing pheromone in the heuristic space is taken to be identical to choosing pheromone in a solution space.

This, however, ignores the reality that the heuristic space is a different kind of space to the solution space and that they are not directly comparable [51]. Therefore this research of comparing the effect of different formulations of pheromone maps is justified as it would properly contextualise how the heuristic space could be searched by an ant algorithm in the most appropriate way.

2.6.3 Justification for Hybridisation of Pheromone Maps

The hybridisation of hyper-heuristics is not something alien to the field. Several attempts have been made to hybridise different elements of hyper-heuristics.

One example made use of the combined efforts of a generation constructive and selection constructive hyper-heuristic to great effect in solving a vehicle routing problem [52]. Another made effective use of all four types of hyper-heuristics in an inter-cell scheduling problem [53]. The scope of hybridisation was even extended to use hybrid hyper-heuristics in automating the design of particle swarm algorithms [54].

So the potential for hybridisation does exist within the field of hyper-heuristics. However, as pointed out by Beckedahl and Pillay [51], heuristic spaces and solution spaces are separate and distinct in terms of their operations. The investigation of the multiple types of pheromone maps also opens up a line of inquiry into whether differentiations need to be made between the maps. More specifically, if the ant-based hyper-heuristics with different pheromone maps are distinct, with their advantages and disadvantages, then there may yet be potential in attempting to use all of them together as opposed to individually. Hence, the justification for attempting to combine the ant-based hyper-heuristics, with their distinct pheromone maps, together is an extension of existing lines of thought for hybridising hyper-heuristics.

2.7 Problem Domains

This thesis concerns itself with four different problem domains and presents the relevant background information pertinent to each domain. These domains are the one-dimension bin packing problem (1BPP), the quadratic assignment problem (QAP), the capacitated vehicle routing problem (CVRP) and the movie scene scheduling problem (MSSP). The first three are well known NP-hard problems and are commonly used in discrete combinatorial optimisation research. The last problem is a newly developed discrete combinatorial optimisation problem.

This approach enables the different hyper-heuristics to be assessed in both relatively familiar and unfamiliar domains to quantify the performance of the use of ant algorithms by the hyper-heuristics. The goal is not to derive some optimal hyper-heuristic, but rather to study the hyper-heuristics in isolation as they cannot be compared.

Finally, this section also contains summaries of the literature regarding existing solution techniques that have been applied in these domains including hyper-heuristics where applicable. This research, however, does not focus on the state of the art methods for comparison; it merely provides context to the research.

2.7.1 1BPP

The one-dimensional bin packing problem (1BPP) concerns itself with the task of packing several items, n, into x number of bins with all bins typically of the same capacity [6]. It is a minimisation problem to reduce the number of bins used to pack all of the items.

In terms of solutions, there are several offline and online heuristics that have been used to solve bin packing problems [55]. Some of the more popular ones include the Next Fit (NF), First-Fit (FF) and Best-Fit (FF) heuristics. Despite the many heuristic methods, exact techniques do exist like the Bin Completion algorithm created by Korf [56] and then later improved [57]. Despite these improvements, there remains no universally perfect method for solving the BPP.

More recently, and of interest to this research, is the use of GP to automate the design of bin-packing heuristics [58]. This represents a large contribution to the field as the authors were successfully able to develop competitive packing heuristics with the GP technique, establishing that hyper-heuristics can be viable for this domain.

2.7.1.1 Problem Definition

The basic form of the 1BPP is to pack n items of a fixed size into m number of bins of a given fixed capacity C. The solution is optimised based on the number of bins needed to contain all of the items with the optimal solution using the fewest number of possible bins.

An alternative fitness function, however, is used in this research. The fitness function [59] is presented in Equation 2.10.

$$Fitness = 1 - \left(\frac{\sum_{i=1}^{n} (\frac{\sum_{j=1}^{m} v_j x_{ij}}{C})^2}{n} \right) \tag{2.10}$$

where n = number of bins, m =number of items, v_j = size of the item j, $x_{ij} = 1$ if piece j is in bin i and 0 otherwise. Finally C = bin capacity.

This function, Equation 2.10, prioritises minimising the wasted space in each bin, favouring bins that are nearly full or full. This avoids the issue of large fitness plateaus that might arise if just the number of bins was used as the fitness value as two solutions could use the same number of bins but have differing levels of fullness with the one minimising wasted space being preferable.

2.7.2 QAP

The quadratic assignment problem or QAP is a combinatorial optimisation problem that revolves around the optimisation of the logistical flow between facilities or factories and where they are located [60].

With the utility of such a problem in many industrial and commercial applications, there are tailor-made solutions for the problem like the FANT [5]. However, there is interest in hyper-heuristic solutions to this problem as well such as [61]. This has even led to the inclusion of the QAP in the HyFlex benchmark set for hyper-heuristics [62], enabling greater research of the application of hyper-heuristics to the QAP. Despite its inclusion in the Hyflex benchmark, the QAP has yet to be properly studied in the context of hyper-heuristics research, something this research is aimed to remedy.

In terms of solutions, exact solutions do exist for the QAP problem but these struggle once the problems grow in size [63]. Additional techniques like

TS, hybrid GAs and branch-and-bound algorithms have all found great success in solving QAP problems [64] although research remains ongoing for the largest QAP instances which can be difficult to solve with either meta-heuristic or exact methods.

2.7.2.1 Problem Definition The QAP problem consists of two elements: locations L and facilities F. L and F are sets such that $L \in \{L_1, ...L_x\}$ and $F \in \{F_1, ...F_x\}$. These sets refer to lists of facilities and locations to place said facilities on. Each facility has the potential to have a logistical flow with other facilities, which refers to the movement of industrial outputs or inputs between facilities. The goal of this problem is to assign the set of facilities to locations with each facility assigned to a single location such that flow amongst facilities is minimised. It can be formalised as:

$$f(\pi) = \sum_{i,j=1}^{n_x} d_{ij} f_{\pi(i)\pi(j)} \tag{2.11}$$

where d_{ij} is the Euclidean distance between locations i and j and f_{hk} describes the amount of flow between facilities h and k.

2.7.3 Capacitated Vehicle Routing Problem

Vehicle Routing Problems (VRP) are a class of problems that involve the scheduling of routes of different vehicles such that several customers can be satisfied whilst at the same time minimising the travel distance of all of the vehicles involved [65]. VRPs are especially relevant because of their real-world applications in a variety of industrial and commercial settings (like food delivery, passenger transportation etc) and their study, even more so, given the difficulty of solving these problems [66].

Of particular interest is the CVRP which is the capacitated VRP that adds the additional condition that vehicles have a specific carrying capacity and therefore, no vehicle may operate a route that exceeds its capacity [67].

In terms of hyper-heuristics, there are some relevant research papers. In [68], the authors made use of grammatical evolution (GE) to evolve heuristics that were then applied to a capacitated vehicle routing problem. GE was used to generate heuristics for application to the problem, with the grammar evolving to represent construction heuristics for the CVRP. The particular use of GE was to overcome a problem with standard GP, namely the explosion of the search leading to increasingly meaningless output.

Sim and Hart, demonstrated that a combined generative and selection hybrid hyper-heuristic could be applied to vehicle routing problems with great success [69]. Their techniques highlight that the combination of heuristic methods can be used, even on relatively difficult problems, to produce good solutions in a reasonable time.

More recently, the CVRP has been used with a generation perturbative hyper-heuristic [70]. In this case, GE was used to combine basic sets of operations and components into perturbative heuristics that were applied to various CVRP instances to great effect.

2.7.3.1 Problem Definition The basic VRP is typically modelled as the graph $G = (V, A)$ where $V = \{v_0, v_1, .., v_n\}$ which indicate a number of nodes, representing customers or clients, and the initial vehicle dispatching depot, v_0 and $A = \{(v_l, v_k), v_l, v_k \in V, l \neq k\}$ represents the links that connect each customer with each link having an associated cost $d_{l,k}$. The CVRP is an extension of the basic VRP with the added condition that vehicles have a specific carrying capacity and therefore, no vehicle may operate a route that exceeds its capacity [67]. The fitness of a given solution is the total cost of all of the routes provided no route violates any of its constraints.

2.7.4 MSSP

The MSSP was not entirely derived in isolation. The problem relates to a broad category of similar entertainment scheduling problems. An early example is the rehearsal problem which involved the scheduling of musicians who need to rehearse pieces of music [71]. The ordering of the musicians to pieces is similar to that of actors to scenes. The authors' used a model checking procedure. This was extended to unequal length musical pieces and a two-stage method was then applied [72].

One of the earliest examples of a movie scheduling problem started with fixed length scenes [73]. This study focused entirely on variable actor wages with later authors extending this by applying GA to this problem which produced better results [74]. These early studies did not consider variable-length scenes which is an issue for real-world applications.

Later work increased the complexity by considering different scene durations and different actor wages [75]. The authors made use of a dynamic programming algorithm although it required bounding to function optimally. Typically, a modern movie production requires scheduling at multiple locations.

Naturally scheduling of scenes in different locations was be next [76]. This required additional complexities like transfer costs and transfer times which were factored into the scheduling process. The authors initially made use of a tabu-search method and a particle swarm optimisation (PSO) method. They extended the work to include ant colony optimisation (ACO) [77]. This formulation extended the MSSP further by mirroring the realities of real-world movie production. What is apparent when one looks at the development of the MSSP is that the general trend is towards increasing the complexity of the models and their definitions to better replicate the actual conditions of movie production. No model can perfectly replicate the exact conditions of movie production but some models can get close enough to serve as useful approximations and as such this will be the basis of the MSSP definition considered here.

Given the relative recency of the MSSP as a problem domain for study, there have been relatively few studies that make use of it as a problem domain. A variety of techniques from PSO and GA, to model checking and TS, have been applied. In terms of hyper-heuristics, however, there have been no studies that make use of the MSSP as a problem domain. Therefore making use of the MSSP in this research represents, on its own, a small but novel contribution to the field.

2.7.4.1 Problem Definition In this section, the MSSP is formally defined with its mathematical model, parameters, and inputs. The definition presented here is based on the formulation that extended the model in prior work [78].

The MSSP contains the following elements: the set of n scenes S, the set of m locations L, and the set of o actors A. All scenes must be scheduled only once with no scenes overlapping each other and every scene assigned to only one location. Scenes may be assigned to the same location but this must be at different times. At least one actor must be assigned to a scene although every actor can be in many non-current scenes.

There are several secondary variables required for defining the MSSP:

- W: This defines a set of daily actor wages. They are paid for each day of production they are involved in including downtime between two scenes. Each actor's daily wage is in the range of [50,100].
- D: This defines the duration of each scene in days. Every scene once scheduled will be fully completed. Each scene's duration is in the range of [1,10] days.
- O: This refers to the location assigned to scene i. Each scene is assigned a randomly determined location out of the list of locations L. Every location has an equal probability of selection and every scene must be assigned a location.
- T_{xy}: This variable is a matrix of transfer times (in days) between different locations. The transfer time to move from location x to x is 0. Each value is in the range [1,10] days.
- C_{xy}: This variable is a matrix of transfer costs between different scenes. The transfer cost to move from scene x to x is 0. Each value is in the range [100,999].
- AS: This quantifies the assignment of o actors to scene i. For each scene, a randomly shuffled list of all of the actors is generated. A random number of actors, in the range of [1,n_s], are removed from this list. The remaining actors are then assigned to scene i.

The costs of scheduling scenes are derived from two components: transfer costs between scenes and wages paid to the actors. The task of MSSP therefore is to order the set S into a permutation R such that the costs of scheduling the scenes is minimised:

$$min \left(\sum_{m \in R} \sum_{n \in R} \sum_{p \in A} \mu_{mnp} W_p + \sum_{n \in R} \sum_{m \in R} \lambda_{mn} C_{mn} \right) \tag{2.12}$$

Equation 2.12 gives the mathematical formulation of the MSSP. The cost function is made up of two components. The first, $\mu_{mnp}W_p$, consists of the time interval of a given actor p between scenes m and n multiplied by their daily wage. This is done for all actors across all of their scenes. The second component consists of $\lambda_{mn}C_{mn}$ which establishes the total cost to move between all scenes where λ_{mn} is set to 1 if the scenes m, n are adjacent in R, the scheduling order and 0 otherwise. These are the two major cost components of the movie produced. The task of MSSP, therefore, is to order the set S into a schedule R such that the costs of scheduling the scenes are minimised.

2.8 Summary

This chapter has laid out the foundations for the concepts and material that need to be understood for subsequent chapters. The chapter provided a brief introduction to combinatorial optimisation. Greater detail was provided to explain the functioning of ant-colony optimisation and the broad family of hyper-heuristics. The chapter presented a critical analysis of why this research is worth doing. The chapter concluded with descriptions of the problem domains considered in this research and any appropriate background information pertinent to them. The following chapter presents the research methodology used in this research to achieve its objectives.

CHAPTER 3

Methodology

3.1 Introduction

This chapter provides the outline for the research methodology considered in this thesis. This methodology is used to meet the research objectives outlined in Section 1.2 of Chapter 1. The rest of this chapter is structured as follows. Section 3.2 provides an overview of pertinent methodologies to this research. Section 3.3 discusses how the chosen research methodology will be able to meet the research objectives outlined in Section 1.2. Section 3.5 provides an outline of how comparative analysis techniques are used to compare the different types of pheromone maps and their effect on ant-based hyper-heuristics. In Section 2.7, the domains considered in this research are presented. Section 3.4 describes the benchmark datasets used for each problem domain. Section 3.5.2 outlines the various experiments that are conducted to meet the research goals. Section 3.6 lists the pertinent technical specifications of the hardware and software used in this research. Finally, Section 3.7 provides a summary of the entire chapter.

3.2 Research Methodologies

The choice of a research methodology for research in the field of computer science is not simple nor straightforward [79]. This is because according to Demeyer [79], computer science has a multidisciplinary origin, with many different fields contributing their part to its development, with mathematics and engineering playing pivotal roles. In particular, mathematical proofs and engineering methods based on experiments and measurements are prevalent depending on the type of research being conducted [79].

In terms of pertinent research methodologies, Oates [80] presents the action research and design and creation research methodologies. Johnson [81] presents the proof by demonstration and empiricism methodologies. Each of these methodologies has its specific uses which are discussed below.

3.2.1 Action Research

The action research methodology is widely used in the fields of psychology, sociology and the medical sciences with later adaptations enabling its use in computer

science and information systems [82]. Action research consists of a five-phase cycle [83]. These phases consist of problem identification, planning, and implementation of actions based on planning and examining the outcomes of the actions with a goal of learning and improvement of subsequent phases.

3.2.2 Design and Creation

Whereas action research evolved from existing sciences (psychology and sociology) and was then adapted for computer science, the design and creation research methodology is more suited for computer science research tasks that typically focus on the use of the computer (algorithms and computer systems) to solve a problem [80]. The key to this methodology is the development of a computer artefact that solves the research problem with iterative assessment and improvement steps to refine the artefact over time.

3.2.3 Proof by Demonstration

Proof by demonstration, as a research methodology, is one based on iterative refinement of an algorithm to produce the desired solution [84]. As the approach or algorithm is developed, evidence of the success or failure of the approach is used to guide the development and refine the approach until it meets the desired research goals. In this sense, the proof by demonstration methodology aims to demonstrate, via the creation of a software artefact, that a given approach can provide a solution to a problem posed by the research.

In the case of this research, the proof by demonstration is applicable as it is used to demonstrate that all four types of hyper-heuristics can employ ant algorithms to search the heuristic space of a wide variety of problems and different hyper-heuristics.

3.2.4 Empiricism

Johnson described the empiricism research methodology as it could be applied to research in computer science [81]. The core of the research method is to generate and then test a hypothesis that encapsulates the core ideas being investigated by the research. In this way, the empirical methodology can be used to meet some of the research objectives defined in Section 1.2 as these objectives relate to hypothesises posed about the nature of the different pheromone maps.

3.3 Hybrid Research Methodologies

The primary goal of this thesis is to examine the effect that different formulations of the pheromone map will have on ant-based hyper-heuristics. However, this process will also require the development of algorithms that use ant algorithms to drive the hyper-heuristic process. As a result, a hybridisation of the

aforementioned research methodologies will be needed. The proof by demonstration methodology is used to demonstrate that ant algorithms can be employed by all four types of hyper-heuristics with the empirical methodology used to then assess the effects that the different pheromone maps have when used by the ant algorithms in all of the hyper-heuristics. This section is dedicated to explaining how the research objectives laid down in Section 1.2 will be met through the research methodologies.

3.3.1 Objectives One to Four

The first four objectives of this research are presented in Section 1.2. They concern themselves with the development of ant-based hyper-heuristics for the four hyper-heuristic tasks. These objectives principally concern themselves with the feasibility of using ant-based methods for hyper-heuristics with the ultimate aim of this research being to improve the application of ant-based methods in hyper-heuristics.

To achieve these objectives, a hyper-heuristic ant colony optimisation algorithm (HACO) is designed and implemented. This HACO algorithm is further specified by the type of hyper-heuristic the HACO algorithm is used to drive. This is given as:

- Selection Constructive: HACO-SC
- Selection Perturbative: HACO-SP
- Generation Constructive: HACO-GC
- Generation Perturbative: HACO-GP

Following the proof by demonstration methodology, the four HACO algorithms will be evaluated with pre-determined assessment criteria to determine if they meet the required outcomes. If the algorithms fail to do so then they should be analysed and those insights gleaned from the analysis should be used to refine the algorithms until they produce the desired outcome or no further improvement is possible.

1. Create an Initial Approach: Create an initial implementation of the HACO algorithm.
2. Define Evaluation Criteria: The individual HACO algorithms are evaluated by examining their performance in terms of optimality, defined by problem-specific objective functions in Section 2.7, and generality, defined by a metric presented in Section 3.5. Each HACO algorithm is evaluated on two problems. The HACO-GC is evaluated on 1BPP, the HACO-GP on CVRP and the HACO-SC and HACO-SP are evaluated on the QAP. The other problem for each hyper-heuristic is the MSSP. Each HACO will be assessed against the other HACOs with different pheromone maps in the same hyper-heuristic. The best-known solutions will be used in each domain as a baseline of the algorithm's performance. Their inclusion will help establish an idea of the algorithm's relative successes or failures in each domain.

The QAP, 1BPP and CVRP are all well known combinatorial problems that have a history of use in previous hyper-heuristic research. Their inclusion also provides a variety of different problem types for assessment. The MSSP problem, by contrast, was selected as it is a relatively new problem and as such could also demonstrate the capacities of ant-based hyper-heuristics for a new domain.

3. Development: Analyse, design, implement and evaluate each of the HACO algorithms. Each algorithm variant is expanded in detail for each of the hyper-heuristics. The evaluation criteria are defined in Step 2.
4. Refinement: If the HACO algorithms fail to reach the desired solution, as determined by comparisons with known or existing good solutions, the following changes could be considered:
   - The parameters of the HACO algorithm.
   - The components, operators or low-level heuristics of the HACO algorithm, are dependent on the given hyper-heuristic in question.
   - The probabilistic ant decision method: tournament selection, roulette wheel, elitism, etc.
5. Repeat steps 3–5 until the desired outcome is acquired or no longer possible.

3.3.2 Objective Five

Objective five in Section 1.2 concerns itself with extending the existing algorithm framework so that HACO algorithms using the three types of pheromone maps can be used in a single hyper-heuristic. In this way, a hybridisation of the HACO algorithms is achieved. To achieve this, an extension to the existing HACO algorithm is designed and implemented. This is the hyper-heuristic ant colony optimisation hybrid (HACOH) algorithm. The HACOH algorithm is further specified by the type of hyper-heuristic that it is. This is given as:

- Selection Constructive: HACOH-SC
- Selection Perturbative: HACOH-SP
- Generation Constructive: HACOH-GC
- Generation Perturbative: HACOH-GP

The process for reaching this objective is as follows.

1. Create an Initial Approach: Create an initial implementation of the HACOH algorithm. This implementation is based on the approach used for the HACO algorithm.
2. Define Evaluation Criteria: The individual HACOH algorithms are evaluated by examining their performance in terms of optimality, defined by problem-specific objective functions in Section 2.7, and generality, defined by a metric presented in Section 3.5. Each HACOH algorithm is evaluated on two problems. The HACOH-GC is evaluated on 1BPP, the HACOH-GP on CVRP and the HACOH-SC and HACOH-SP are evaluated on the QAP. The other problem for each hyper-heuristic is the MSSP. Each HACOH will be assessed against the non-hybrid HACOs.

3. Development: Analyse, design, implement and evaluate each of the HACOH algorithms. Each algorithm variant is expanded in detail for each of the hyper-heuristics. The evaluation criteria are defined in Step 2.
4. Refinement: If the HACOH algorithms fail to reach the desired solution, the following changes could be considered:
   - The parameters of the HACOH algorithm
   - The components, operators or low-level heuristics of the HACOH algorithm, dependent on the given hyper-heuristic in question.
   - The hybridisation scheme control parameters.
   - The hybridisation perturbation operator: random permutation operator or gradual mutation operator.
   - The probabilistic ant decision method: tournament selection, roulette wheel, elitism, etc.
5. Repeat steps 3–5 until the desired outcome is acquired or no longer possible.

3.3.3 Objective Six

The sixth objective defined in Section 1.2 relates to one of the fundamental objectives of this research. Namely, determining the effect of using different pheromone maps for the different types of hyper-heuristics. The prior objectives concerned themselves with actually building the algorithms but this one relates to their serious evaluation. Hence the use of the empirical methodology as previously defined.

To this end, the process for reaching this objective is as follows:

1. Generate the hypotheses: There are several hypotheses that this research hopes to assess. These hypotheses were determined through examination of the results of prior ant-based hyper-heuristics conducted as part of this thesis. They are:
   (a) The type of pheromone map will significantly affect the performance of the ant-based hyper-heuristic.
   (b) 1D pheromone maps work better for selection hyper-heuristics than 2D or 3D pheromone maps.
   (c) 3D pheromone maps work better for generation hyper-heuristics than 1D or 2D pheromone maps.
   (d) The hybridisation of HACO will work better for an ant-based hyper-heuristic than a non-hybrid equivalent.
2. Perform Experiments: Many experiments will be performed using each type of ant-based hyper-heuristic across all of the specified problem domains. The results of the experiments will be compiled and assessed with statistical tests outlined in Section 3.5.
3. Evaluate Hypotheses: The results of the tests will then be used to affirm or reject the hypotheses proposed in this research.

3.4 Benchmark Datasets

This section describes the benchmark datasets used for each of the problem domains. The best-known values for the domains, where applicable, will be presented in Section 10 during the presentation of the results.

3.4.1 1BPP

The datasets sets used in this study described in Table 3.1.

Table 3.1: Datasets in the Uniform and Hard 1BPP Benchmark

1BPP Datasets	Number of Instances	Number of Items	Bin Size
u120	20	120	150
u250	20	250	150
u500	20	500	150
u1000	20	1000	150
Hard	10	200	100000

These datasets are taken from the Falkenauer [85] and Scholl [86] benchmarks. The data for the benchmarks was taken from [87] and [88] respectively. The number of items and the size of each bin are provided for the instances that fall within each of the datasets that make up the benchmark. These sets were chosen based on existing literature for generation hyper-heuristics [58]. In terms of these datasets, the instances from the Hard dataset are regarded as particularly difficult as the size of the items is large and many items have to be packed quite well together to minimise wasted space. With regards to the other datasets from the Uniform benchmark, the instances scale in difficulty as the number of items needing to be packed increases. The easiest instances are found within the u120 dataset with the hardest instances being found in the u1000 where many items have to be packed into relatively small bins.

3.4.2 QAP

All of the QAP data was acquired from QAPLib [89]. The QAPLib is a repository that combines many of the known QAP benchmarks into a single location.

Table 3.2 provides describes the characteristics of the QAP instances that have been used in this study. The number of facilities and locations are indicated for each instance.

As a quadratic problem, the number of facilities and locations will always be equal. In general, QAP problem instances become more difficult as these two values increase. Very small problems, like chr12a for instance, can even be solved with brute force as the solution space is small in comparison to the larger

Table 3.2: QAP Benchmark Instances

Instance	Number of Facilities	Number of Locations
chr12a	12	12
chr12b	12	12
chr12c	12	12
chr15a	15	15
chr15b	15	15
chr15c	15	15
chr18a	18	18
chr18b	18	18
chr20a	20	20
chr20b	20	20
chr20c	20	20
chr22a	22	22
chr22b	22	22
chr25a	25	25
els19	19	19
had12	12	12
had14	14	14
had16	16	16
had18	18	18
had20	20	20
kra30a	30	30
kra30b	30	30
kra32	32	32
scr12	12	12
sko42	42	42
sko49	49	49
sko56	56	56
sko64	64	64
sko72	72	72
sko81	81	81

problems like sko42 and larger. These problems provide a wide range of potential challenges for the hyper-heuristics and should demonstrate their capacities across a range of potential problem instances.

3.4.3 CVRP

This study makes use of data taken from CVRPLIB [90]. The instances are represented in Table 3.3. In the above table, n is the number of vertices that need visiting, K is the number of vehicles and Q is the capacity of each vehicle. These instances were chosen to reflect a wide variety of CVRP instances of varying

Table 3.3: CVRP Benchmark Instances

Instance	**n**	**K**	**Q**
A-n32-k5	31	5	100
A-n33-k5	32	5	100
A-n33-k6	32	6	100
A-n34-k5	33	5	100
A-n36-k5	35	5	100
A-n37-k5	36	5	100
A-n37-k6	36	6	100
A-n38-k5	37	5	100
A-n39-k5	38	5	100
A-n39-k6	38	6	100
A-n44-k6	43	6	100
A-n45-k6	44	6	100
A-n45-k7	44	7	100
A-n46-k7	45	7	100
A-n48-k7	47	7	100
A-n53-k7	52	7	100
A-n54-k7	53	7	100
A-n55-k9	54	9	100
A-n60-k9	59	9	100
A-n61-k9	60	9	100
A-n62-k8	61	8	100
A-n63-k9	62	9	100
A-n63-k10	62	10	100
A-n64-k9	63	9	100
A-n65-k9	64	9	100
A-n69-k9	68	9	100
A-n80-k10	79	10	100
M-n101-k10	100	10	200
M-n121-k7	120	7	200
M-n151-k12	150	12	200
M-n200-k16	199	16	200
M-n200-k17	199	17	200

sizes. In terms of the problem difficulty, the CVRP instances scale in difficulty as the quantity of the problem components increase. The smaller problems, like A-n32-k5, with five routes and thirty-two vertices, are much easier to solve than the larger ones like M-n1210-k7. As the number of routes increases, it becomes more difficult to determine the optimal solution as the potential number of configurations for each route grows, so problems with a similar number of vertices but a different number of routes can differ significantly in their difficulty.

3.4.4 MSSP

This research makes use of the dataset generated in prior research [78]. This establishes continuity with that prior research in terms of dataset reuse and also obviates the need to generate new instances as well. As such, it makes use of the four different problem classes, each with five problems, for a total of twenty instances. Each of the instances within each class has similar objective values in the sense that all problems of the same class should have optimal values that are very similar. Across problem classes, however, is where larger differences in the objective values will be noted. The description of the problem classes is given in Table 3.4:

Table 3.4: MSSP Benchmark Instances by Class

Class	**0**	**1**	**2**	**3**
Scenes (S)	10	25	50	100
Actors (A)	5	10	30	60
Locations (L)	3	5	10	15

3.5 Comparative Analysis and Assessment

To achieve the objectives of this research, a comparative analysis will be conducted on the results of the experiments conducted with the HACO and HACOH algorithms. This entails conducting several experiments in the domains described in Section 2.7. These experiments will be assessed using the metrics and methods described in Section 3.5.1.

3.5.1 Assessment Metrics

Assessment of hyper-heuristics needs to be holistic to get an accurate picture of how the hyper-heuristic is performing, not merely as a solution to the underlying problem, but also as a hyper-heuristic itself.

3.5.1.1 Optimality Assessment of a hyper-heuristic's optimality refers to the quality of the solutions that the hyper-heuristic can produce. This is typically measured with the fitness value of the solutions produced by the hyper-heuristic and of course, will vary amongst the different problem domains. Importantly, the assessment of optimality happens between instances in a given problem domain, and not across that domain. This means that when assessing the optimality of a given hyper-heuristic, the comparisons are drawn specifically against the performance of the hyper-heuristic in each problem instance. This way of assessing

the hyper-heuristics provides a high level of granularity that can demonstrate trends in the hyper-heuristic's performance that might otherwise be hidden by a higher level of abstraction.

For each of the domains, this assessment will make use of the fitness value for those domains as the basis for the optimality comparison. However for benchmark sets with a large number of instances, like the 1BPP, a fitness ratio, F_R is calculated. This represents the average quality of solutions over a certain number of instances. The reason for this is to allow for improved clarity in presentation. The ratio, F_R, is calculated as follows:

$$InstanceAverage = \frac{\sum_{i=1}^{I} r_i}{I}$$

$$InstanceRatio = \frac{InstanceAverage}{z} \tag{3.1}$$

$$F_R = \frac{\sum_{i=1}^{m} InstanceRatio_i}{m}$$

The $InstanceAverage$ is the mean value of the algorithm's performance over I runs. This is then divided by the optimal solution for that instance, z. Finally, the average of all of these $InstanceRatios$ is averaged over several instances, m. In terms of interpretation, the smaller the value, the more optimal the algorithm as values close to 1.0 represent the algorithm producing near-optimal solutions.

3.5.1.2 Generality The other major assessment metric for a hyper-heuristic is generality. Unlike optimality, generality is assessed across all the instances of a given problem domain because generality reflects an aggregated assessment of how well the hyper-heuristic performs over many instances.

To assess the various algorithms in terms of their generality, a generality metric, standard deviation of distances or SDD, is employed [91]. It is formulated as:

$$SDD(H) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{3.2}$$

SDD is designed to assess the performance of a hyper-heuristic of several problem instances N. The lower the value the better the score. This metric will be useful in understanding the degree to which different pheromone maps will be able to generalise across problems and domains. Generality is one of the goals of hyper-heuristics and hyper-heuristics that have improved generality are preferable to ones with reduced generality [91].

3.5.1.3 Pheromone Map Analysis To complete the analysis of the hyper-heuristics, it is important to understand how the different pheromone maps

behave with respect to the hyper-heuristic process. The concentration and distribution of pheromone in the map will be demonstrative of the characteristics of that map type. To that end, visualisations of these pheromone maps as heatmaps will aid in the analysis. The heatmaps will be generated using the tool detailed in [92].

3.5.1.4 Existing Heuristics

While it is not the primary focus of this research, it can be important to contextualise the performance of the ant-based hyper-heuristics in some way. To this end, the results of the hyper-heuristics will be compared against existing heuristics in each of the domains. The comparison against existing heuristics provides a solid grounding point for demonstrating whether the different pheromone maps are improving or not improving the success of the hyper-heuristic.

3.5.2 Experiments

Two primary experiments will be conducted during this research. This section provides an overview of the experiments which are described in greater detail in Chapters 6–8.

The first set of experiments to be conducted involves testing each of the HACO algorithms on their respective problem domains with 1D, 2D and 3D pheromone maps respectively. Each of these simulations will be conducted with 30 runs. During each run, the hyper-heuristic will attempt to search the respective heuristic space to either generate heuristics that are applied to a problem or produce a selection of heuristics.

The second set of experiments involves the HACOH algorithms in the same domains as in the first experiment. The major difference between these two experiments is that executing the HACOH algorithms is a two-phase process. The first phase of the process involves determining how the different pheromone maps will be used in the second phase which executes on the full benchmark set in the problem domain. This is described in greater detail in Chapter 9.

3.5.3 Parameter Sensitivity

The HACO algorithm has several parameters. Most of them relate directly to the operation of the ant algorithm itself but some relate more to its use in experiments. The control parameters are p, the rate of evaporation and α, a desirability weight for pheromones. These two parameters are adjusted by the algorithm itself and so do not require parameter tuning. There are two parameters w_1 and w_2 which are pertinent for the 1D HACO algorithm as well which is also adjusted internally by the algorithm. These parameters and their configurations are described in greater detail in Section 5.7.

Another parameter p_l, the path length, is more specific to each of the different hyper-heuristics and its configuration is described in the chapter about the appropriate hyper-heuristic, Chapters 6–8.

In terms of the remaining parameters for the experiments, the number of ants n_k and the number of iterations are most important. The remainder of the control parameters is dynamically adjusted as the algorithm executes. Therefore a process is used to determine good values for n_k and the number of iterations for a fair comparison of the HACO and HACOH algorithms.

More specifically, a 2D HACO algorithm was executed 30 times with different values for n_k and the number of iterations to assess how well the algorithm performed with these values. This is a grid-search technique of different configurations for the algorithm. As computing resources are not unlimited, the range of values chosen for both variables is

- n_k: [5,10,20,50,100]
- Number of iterations: [50,100,200,500,1000]

These values provide a large enough coverage of different experimental configurations without requiring excessive computing resources. Optimality considerations are not the only factor to weigh when deciding on the parameters. The computational runtimes are also important to consider. In an ideal world, the experiments could be run with the largest possible configurations, but in practice, there are trade-offs to consider concerning algorithm runtimes as well.

The number of runs is fixed at 30 to provide a large sample of data for statistical testing. As the purpose of this research is the comparison between HACO algorithms using different pheromone maps, the HACO algorithms must be tested with the same experimental conditions for the comparison to be fair. To that end, the 2D HACO is used for this process to establish a baseline assessment of the algorithm's capacities. As the conditions will be the same for all types of HACO, any differences that result in the experiments will be primarily due to the influence of the type of pheromone map, and not that specific algorithm's parameters.

This process will be performed for each type of hyper-heuristic with both the optimality and runtime of the algorithm taken into consideration for the analysis. Additionally, correlation coefficients [93] will be calculated for the comparisons between fitness and the number of iterations and n_k respectively.

The input data for this process will be taken from the MSSP domain. Specifically, the C_S_2_I0 was chosen because it is one of the larger instances in the MSSP domain and should serve as a sufficiently difficult problem for the various algorithms. Its size and complexity in relation to the other instances make it an ideal candidate for use in the parameter tuning process. The MSSP domain is used across all four parameter tuning processes because it allows for a standardised comparison of the parameter tuning results as the process uses the same data despite the type of hyper-heuristic being applied.

This research is not aiming to produce the best possible values for the given problem domains and as such, it is unnecessary to tune the algorithm for each different instance. Rather it is to assess algorithm behaviours under a broad spectrum of different experimental configurations. These results will then be assessed to choose a final configuration for the full experiments. These smaller-scale

tests are meant to provide rough approximations of the algorithm performance so that larger and more comprehensive tests can be performed.

Finally, the values that are determined by this process will be used by the HACOH algorithm as that algorithm uses HACO algorithms that are otherwise indistinguishable from the normal ones, and do not require special configuration to work.

3.5.4 Statistical Tests

Due to the nature of the research in this thesis, statistical testing is needed to properly assess the validity of the results of the experiments. Based on guidelines for comparing stochastic algorithms [94], the recommended approach is to perform a multiple comparison test. This research makes use of the Friedman test [95] as it is a widely used statistical test and appropriate for the comparison of multiple algorithms. In addition, a post-hoc analysis procedure is required to refine the significance of the testing results.

In this case, the use of the Mann-Whitney U Test[96] is then used to examine where the differences between algorithms might occur. The reason for this is that samples are all from independent runs of the algorithm; one set of algorithm executions has no bearing on the outcomes of execution of another.

As these tests make use of multiple comparisons, the Bonferroni correction method is applied to determine if the differences between algorithms are statistically significant [97]. The purpose of this testing procedure is to establish the statistical significance of the results of comparing the different types of pheromone maps in the ant-based hyper-heuristics to one another.

Additionally, an effect size metric is calculated for each of the statistical Mann-Whitney U Tests. The effect size is used to quantify the magnitude of the difference between tested groups. It is used in conjunction with the statistical test as alone, it cannot determine if a difference is statistically significant. However, for statistically significant results, it quantifies the magnitude of the difference between the results.

The effect size used in this case is Hedge's g [98,99]. Like Cohen's D, [100], Hedge's g has some issues regarding bias when the samples are small (less than fifty elements). To correct this, a bias correction formula [101] is applied to the Hedge's g calculation. The formula is:

$$\frac{n-3}{n-2.25} * \sqrt{\frac{n-2}{n}} \tag{3.3}$$

where n is the sum of the number of samples from both groups being compared. Finally in terms of the interpretation, Cohen [100], suggests the following:

- Small Effect: An effect size around 0.2 is considered small.
- Medium Effect: An effect size around 0.5 is considered a medium effect.
- Large Effect: An effect size around 0.8 is considered a large effect.

The important aspect to emphasize regarding the effect size is the magnitude. The effect can be positive or negative depending on the differences between the two samples but only the magnitude is important in this case.

3.5.4.1 Statistical Test Procedure The Friedman test is conducted at a 0.05 significance level. This test is used to establish whether there are significant differences across the results of the experiments. For example, testing if there is a statistically significant difference between different HACO algorithms in a domain. This will filter out the instances where the differences are minimal between the methods being tested.

The Mann-Whitney U Test is then applied in a post-hoc analysis procedure to determine which of the comparisons (between algorithms) are specifically significant if they have reached significance under the Friedman Test. This is conducted with a one-tailed test at a 0.05 level of significance.

The hypothesis for this test is as follows:

- The Null Hypothesis (H0): $\mu_1 \geq \mu_2$
- Alternative Hypothesis (H1): $\mu_1 < \mu_2$

In this case, μ_1 and μ_2 represent the mean values of a sample of output results from different techniques 1 and 2 respectively. These tests compare the means of these sets of samples to establish which of the means (and therefore techniques) has the lower mean and thus the better performance on average.

The Bonferroni correction method is applied to these tests to account for the repeated comparisons. The testing procedure compares the 2D pheromone map against the 1D and 3D, as well as the 1D and the 3D. This is to establish which of the maps is optimal. Then, the hybrid (HACOH), is compared against the 1D, 2D and 3D pheromone maps, in the same way, to determine if the hybrid outperforms any of the non-hybrid methods.

3.6 Technical Specifications

For this research, a computing cluster provided by the University of Pretoria was used for running the experiments. The technical specifications of this cluster are 377GB RAM, 56 cores at 2.40GhX (Intel Xeon CPU E6-2680 v4) and 1TB of Ceph Storage. Java 1.8 with the Netbeans 8.1 Integrated Development environment was used for the development of the software. The implementation of the HACO/HACOH algorithms makes use of the concurrent programming methods to improve processing times. Additional statistical tests and calculations were provided by Python and Excel where needed.

3.7 Summary

This chapter presented the methodology to be used to investigate the effect of different pheromone maps on ant-based hyper-heuristics. This chapter also detailed the various problem domains, and their respective datasets, that will be used for the experiments. The next chapter presents the explanation of how different pheromone maps could be formed.

CHAPTER 4

Pheromone Maps for Hyper-Heuristics

4.1 Introduction

This chapter provides an in-depth explanation of how the pheromone mechanism that underlies an ACO algorithm operates and will be modified for use in this research. It also provides details and explanations of how the pheromone maps can be compressed and projected onto different dimensions for use in hyper-heuristics. Section 4.2 details the operation of conventional pheromone spaces. Section 4.3 details how a pheromone map can be projected into the third dimension. Section 4.4 details how a pheromone map can be compressed into a single dimension. Finally a summary is given in Section 4.5.

4.2 Pheromone Mechanism

The basic mechanism of the ACO algorithm is the pheromone map. One of the limitations of ACO has been that it depends on a graph-based representation for the problem it will be applied. The graph-based representation naturally lends itself toward representing the pheromone map which holds the search information gathered by the ants [11]. As the ants perform their search they deposit pheromone on parts of the map to correspond to links in the ant's search path. For the sake of illustration, consider the case of a basic QAP as in the case of Figure 4.1. This applies to any applicable problem for ACO but the use of a simple problem aids in clarifying the relevant principles.

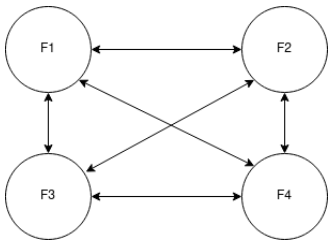

Fig. 4.1: A Simple QAP Problem

This graph has four nodes, F1 to F4, and represents several facilities to pick from selecting which facilities to place. A corresponding pheromone map for this problem, in the solution space, would look like Figure 4.2.

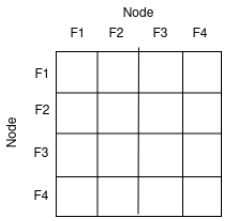

Fig. 4.2: 2D Pheromone Map

This representation is of the solution space and indicates how ants can build a solution for the problem. By traversing the graph and choosing facilities, a solution is created. The best solution emerges over time through the accumulation of pheromone in better regions.

However, this is not directly transferable in the case of a hyper-heuristic that operates in the heuristic space, and not the solution space. Consider a representation of the heuristic space in Figure 4.3.

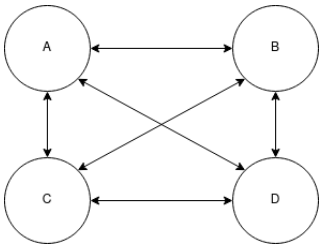

Fig. 4.3: A Simple Heuristic Space

This figure represents a heuristic space for a selection constructive hyper-heuristic task. For the sake of the example, this considers a selection constructive hyper-heuristic but these principles apply to all of the hyper-heuristics with the appropriate heuristic spaces. Each node (A, B, C, D) represents a low-level heuristic that can be applied to progressively build a solution for a given problem. This space differs significantly from the solution space in several ways. Firstly, in a solution-space-based pheromone map, there is semantic meaning to choosing to deposit pheromone at link (i, j) because of the direct corresponding relationship to the solution. The link (F1, F2) is meaningful because it is unique and relates directly to an overall structured solution.

In the heuristic space, however, multiple heuristics can be repeated several times over, with duplicated links appearing in a given solution. For example, a heuristic selection of just four heuristics could look like this,

```
<A,B,A,A,B,C>
```

In this case, the link (A, B) appears twice in the set but at very different places. Therefore attempting to deposit pheromone at this link will simply deposit double the pheromone at one location ignoring that this link appears at different points in the heuristic solution with all the potential significance this difference could apply.

The problem with a 2D representation is that, while it can determine which links (between components) are important, it cannot account for where those links might be in the actual path being constructed. The same link can be found at different points in the path and not necessarily have the same effect for example.

4.3 Map Projection

The solution to the loss of information problem is to add another dimension to this matrix, essentially projecting it into the third dimension. This is depicted in Figure 4.4.

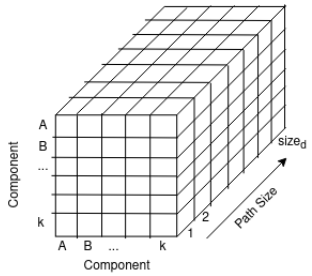

Fig. 4.4: 3D Pheromone Map

In this new projection, the third dimension represents points in the heuristic path being created by the ant and can be divided into layers. Each layer represents a connection between components in the path at that point in the path. So the first layer represents the first link and so on. In this way, both the connection between components and their position in the actual path is accounted for as the ants can differentiate between different links taken at different points in the search.

The limit of this third dimension is dependent on, primarily, the type of hyper-heuristic the HACO algorithm is being used for. For generation hyper-heuristics, the user can define a limit as this will control the size and complexity of the generated heuristics. This limit is referred to as f_{limit}.

For selection hyper-heuristics, the limits are usually based on the size of the problem as they more directly interact with the problem through the use of low-level heuristics.

How this size is calculated generally depends on the problem but usually will refer to the number of nodes, vertices or other components that make up a solution.

An ant algorithm operating with a 2D pheromone map in the heuristic space should be at a disadvantage in terms of its ability to produce good heuristic selections. However, in terms of the search process, this is not so theoretically clear cut.

Specifically, a path generated from a 2D pheromone map can still be used to deposit pheromone on that same map. The issue is that the map cannot meaningfully differentiate between where a link in the path is and where it should be deposited on the map. For example, if the link (A, B) appears in a path several times, a 2D pheromone map cannot differentiate wherein the path this link occurred and thus will accumulate all of the pheromones on a single location on the 2D map.

So gradually over time, the behaviour that would be expected is for the links in the better paths to accumulate pheromone in greater quantities over other links from weaker paths, resulting in a 2D pheromone map that contains information about the best components, but not how those components should be ordered to produce a heuristic.

To clarify this situation, consider the case of an ant that has to construct a path on an example 2D pheromone map after some pheromone has been already deposited.

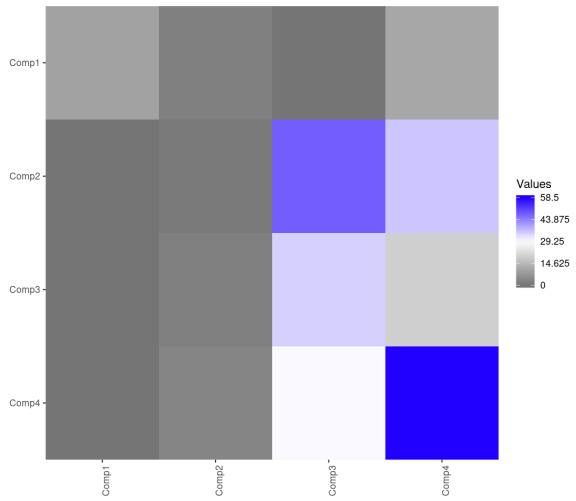

Fig. 4.5: Pheromone Distribution on a 2D Pheromone Map

The pheromone map in Figure 4.5 is represented with the pheromone concentrations given by a colour gradient.

In Figure 4.5, the ant has no idea what order it should visit the links represented by the pheromone deposited on the map. As ACO-based methods are probabilistic, the ant has a probabilistic chance to visit these links in a variety of orders, none of which may end up being the original order that produced the underlying heuristic that was used to deposit the pheromone. If this case is expanded to consider many ants, now it should be seen that these many ants will perform something approximating a local search of these best components. The ants will probabilistically combine these components in such a way that they end up searching these components for combinations that, ideally improve upon past work, but since the ants cannot meaningfully record the structure of their search in the 2D map, their behaviour will always revert to this local searching around the best components.

The 3D pheromone map solves this issue by representing the position of the link in the ant's path in the layer of the 3D pheromone map. Each layer relates to a particular point in the heuristic search for when that amount of pheromone was applicable and in this way, enables a more precise refinement to occur. The reason for this, of course, is that the ants have been transplanted from working in the solution space, for which a 2D map is sufficient, to working in a heuristic space, where those normal conditions no longer apply. This is not to say that the 2D map would be theoretically always inferior to the 3D map.

Theoretically, for small enough problems or problems where the degree of precision in adjusting the pheromone map to reflect heuristic information is minimal, a 2D pheromone map would enable the ants to continue to do their local searching until they found a good heuristic. If a given problem has complicated feasibility conditions for its solutions, the 2D pheromone map should be at a disadvantage as it can only find the best heuristic selection within those feasibility conditions through an extensive search over time, rather than being able to precisely refine the components as would be the case for a 3D pheromone map. In that sense, having a 3D pheromone allows an ant's path through the 3D space as a trajectory to represent the original kind of information as if the ant would have been searching a solution space instead of a heuristic space.

Despite this, the 3D pheromone map comes with drawbacks of its own. Adding a dimension will dramatically increase the search effort required to find good heuristic selections because the space of potential searching has been magnified, by increasing the size of the area needing to be searched.

This has the potential of increasing the overall cost of using the algorithm as compared to the 2D pheromone map. As is the case with the no free lunch theorem, it would be the case that each type of pheromone map would have different uses for hyper-heuristic tasks.

4.4 Map Compression

The general idea around the pheromone space projection, which is going from 2D to 3D, is that the additional projection enables the heuristic space to reflect information that is necessary for a precise refinement of a given heuristic solution. In cases where precise refinement of the heuristic solution is necessary, this capacity, hypothetically, enables a better-suited hyper-heuristic. However for selection hyper-heuristics, constructive and perturbative, specifically there may be issues with the pheromone map formulation that could lead to less than optimal outcomes.

The first aspect is the low-level heuristics themselves. The underlying low-level heuristics for a problem can reflect a wide variety of different methods for building solutions. A heuristic selection is simply a permutation of a set of heuristics that are used to solve a heuristic problem; constructing a solution in this case. In particular, if a low-level heuristic makes use of randomness in any way it can pose a potential problem for any kind of selection hyper-heuristic. This is because two sets of identical heuristic selections could produce two different solutions of differing fitness.

Consider the heuristics A, B and C where A and B are deterministic heuristics in the sense that they will always behave predictably when used in a hyper-heuristic. Heuristic C on the other hand is stochastic in some way. So different executions of heuristic C will have different results each time it is executed. Randomness in low-level heuristics is not necessarily a bad thing as it can increase the degree of exploration during solution construction.

However, the issue arises when the heuristics are used in a heuristic selection such as $< A, B, C >$. As C has some random component, different executions of this selection will produce different solutions of different quality. The effect of this is that feedback mechanisms that are attempting to capture or otherwise use solution information to guide their search mechanisms cannot reliably do so as the single point in the heuristic space can no longer reliably and consistently map to the same point in the solution space. This effect is heightened as the heuristic selection's size grows.

The effect of this problem is further magnified by the addition of multiple solution agents which attempt to construct a population of heuristic selections to facilitate some population-based search of the heuristic space. Multiple ants, each trying to evaluate the merit of their heuristic selections, will now end up polluting their pheromone map with potentially contradictory information. Given this, it may be the case that the 3D pheromone map projection does not offer the best representation of the heuristic space for the specific kind of hyper-heuristic task.

This does mean, however, that an alternative formulation of the pheromone map could offer a representation of the heuristic space that is more amenable to the ACO method. Hence the idea of pheromone space compression, or the 1D pheromone map.

Ultimately the task of a selection hyper-heuristic is to find some optimal selection of low-level heuristics that provide the solution, constructed or perturbed. While the task can be thought of as finding the optimal permutation,

another way to think about the problem is to find an optimal distribution of low-level heuristics for a given problem. In particular, if the low-level heuristics themselves can be subject to something like randomness, then the task becomes more about finding a set of heuristics, that in aggregate, tends to result in the best solution. Consider a 2D pheromone map as given in Figure 4.6.

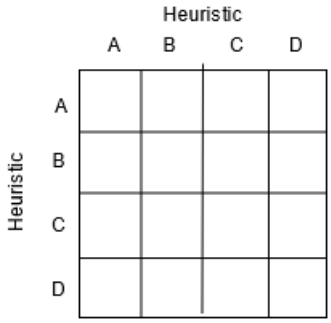

Fig. 4.6: 2D Pheromone Map

This pheromone map represents the initial way of implementing a mapping for the heuristic space in a given selection problem. Figure 4.7 represents a compression of that 2D map into a 1D plane. Only a single row is now needed to represent each heuristic.

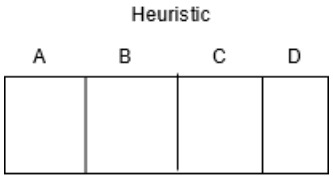

Fig. 4.7: 1D Pheromone Map

In the process of transforming the pheromone map into the one-dimensional plane, what the map represents has changed as well. Ordinarily, a 2D pheromone map represents the linkages between different components in a path towards a solution. The map in its current form now represents the proportion of each heuristic in a given heuristic selection. Each index, therefore, becomes an accumulation of the number of times that the given associated heuristic was included in a heuristic selection. The 1D pheromone map now represents an underlying statistical distribution of the various heuristics in a selection. In this way, the more a heuristic is included in a solution, the more likely that heuristic is to be associated with better fitness. Over time the distribution of pheromone in

this map represents the probability that the given heuristic will be included in a selection.

This adaptation to the pheromone space is inspired partially by the FANT [5]. The FANT methodology will be used in an adaptation of the existing ant algorithms for 1D pheromone spaces. In addition, the use of a single ant typical to the FANT also suits the nature of the task as a single ant minimises the effect of contradictory fitness values from the same heuristic. Even though a single ant is used, this is still within the realm of an ant algorithm in part because the ant operates as a search agent to explore the heuristic space like any other ant, and also because ant colonies have no hard requirements on the number of ants contained within.

These principles are also more broadly applicable to all hyper-heuristics. Generation constructive and generation perturbative hyper-heuristics can incorporate elements into them that facilitate the behaviours that are more associated with selection hyper-heuristics. So the methodology outlined here can be applied to those types of hyper-heuristics as well.

The purpose of the 3D pheromone map is to provide greater precision to the ant-based hyper-heuristic in terms of its ability to precisely capture information about the results of a heuristic selection. This comes at the cost of increasing the computational effort required to perform that search given the larger pheromone map space that is involved. A 3D pheromone map is larger than a 1D pheromone map by a scale of 2 additional dimensions. It will be up to the experiments to determine whether the trade-offs will be worth it.

The 1D pheromone map performs the opposite function. By trading the ability to capture precise information, the 1D pheromone map operates in a much smaller pheromone domain that is much easier to search.

4.5 Summary

This chapter presented an explanation for the different types of pheromone maps used in this research for 1D, 2D and 3D pheromone maps. The next chapter presents the hyper-heuristic ant colony optimisation algorithm.

CHAPTER 5

Hyper-Heuristic Ant Colony Optimisation Algorithm

5.1 Introduction

This chapter introduces the hyper-heuristic ant colony optimisation algorithm (HACO). As discussed in Section 3.3, the HACO algorithm has several variants based on the different types of hyper-heuristics. However, there remains great commonality in the functioning of the HACO algorithm across its variants. Therefore, this chapter presents a generalised HACO algorithm that describes the algorithm without the specific details that are pertinent to the variant and the type of hyper-heuristic. Refer to the later chapters for those specific details. The rest of this chapter is organised as follows. In Section 5.2 a high-level overview of the general HACO algorithm is presented. Section 5.3 presents the pheromone update rules for 1D pheromone maps. Section 5.4 presents the pheromone update rules for 2D and 3D pheromone maps. Section 5.5 describes the general method of path construction in a HACO algorithm. Section 5.6 details the desirability heuristic used in the algorithm. Section 5.7 describes the control parameters of the algorithm and finally a summary is given in Section 5.8.

5.2 HACO General Algorithm Overview

The high-level algorithm of the HACO method is presented in Algorithm 4. This algorithm details a broad overview of how the algorithm functions. Unless otherwise noted, the algorithm also functions the same regardless of the type of pheromone map given to it. This overview broadly applies to all types of the HACO algorithm but some pheromone maps necessitate changes. Those are discussed in Sections 5.2.0.1–5.4.

Algorithm 4 starts with a population of ants that are initially created (with empty paths) between (Lines 2–3). Over several iterations, the ants will construct paths, (Line 9). These paths are evaluated, with a problem-dependent fitness function (Line 11), and that information is used to update the shared pheromone map accordingly (Lines 16–17). The pheromone update procedure is a two-part procedure. First, the existing pheromone will be evaporated according to the appropriate scheme. Then the new values of the pheromone map will

Algorithm 4: High-Level Algorithm

Input: n_k ant colony, it the max number of iterations, ph a pheromone map, p the evaporation rate, α the pheromone desirability, $size_d$ the path limit

Result: S_B the best solution, P_B the best path

```
1  initialise ph with small random values;
2  foreach a ∈ n_k do
3      initialise a
4  i=0;
5  best = inf;
6  for i < it do
7      fitness[] = [n_k];
8      foreach a ∈ n_k do
9          construct a path using Algorithm 7
10     foreach ant a in n_k do
11         fitness[a]=evaluate(a.path);
12         if fitness[a] < best then
13             best = fitness[a];
14             S_B=a.getSolution();
15             P_B=a.getPath();
16     evaporate ph using the Equation 5.1;
17     update ph using the Equation 5.2;
18     update p using the Equation 5.4;
19     update α using the Equation 5.5;
20     i = i + 1;
```

be calculated and the map will be updated. At the end of the algorithm's execution, the best solution found as well as the best path, which was found through a search of the component space, is returned. The component space in this context refers to the space containing all of the possible components (operators and domain attributes) for a given problem.

The general algorithm makes use of an iteration-based stopping criterion that will stop it after a certain number of iterations have been performed. Additional criteria like stopping after no improvements have been observed after several iterations, or after reaching a certain fitness value are possible as well.

The pheromone map is initialised randomly with small random values in the range of [0,1] for the 2D and 3D HACO. In the case of the 1D HACO, each index of the map is set to 1 to ensure each heuristic has the same random chance of selection. This initialisation is needed to make sure that the ants have as wide an exploration of the heuristic space as possible during the first few iterations.

5.2.0.1 1D Modifications Algorithm 4 is generalised to represent HACO algorithms that make use of the 2D and 3D pheromone maps. This is because, at this level of granularity, the distinction between a 2D and 3D pheromone map is minimal. However, some modifications are necessary for the HACO algorithm to use the 1D pheromone map. These modifications are necessary to retain the functioning of the algorithm in the 1D pheromone space but otherwise represent minor modifications to the overall algorithm. In addition, only a single ant is needed for this type of pheromone map based on Section 4.4. The compression of the pheromone map to a single dimension does not fundamentally alter the nature of the ant algorithm that operates on it. The principles of the FANT [5], are applied for this type of pheromone map as they would be for any other in principle. The mechanism of pheromone reinforcement would be the same regardless of the type of pheromone map, but as discussed in Section 4.4, the FANT methodology is adapted for the 1D pheromone map specifically. These modifications are presented in Algorithm 5.

Algorithm 5: High Level Algorithm for 1D Pheromone Map

Input: a ant, it the max number of iterations, ph a pheromone map, α the pheromone desirability, $size_d$ the path limit
Result: S_B the best solution, P_B the best path

```
1  initialise ph to 1;
2  w_1 = 1;
3  w_2 = 1;
4  initialise a;
5  i=0;
6  best = inf;
7  for i < it do
8      fitness = 0;
9      Construct a path using Algorithm 7
10     fitness=evaluate(a.path);
11     if fitness < best then
12         best = fitness;
13         S_B=a.getSolution();
14         P_B=a.getPath();
15     if similarity(current path, best path)=0.85 then
16         w_1 = w_1 + 1;
17     if fitness <best then
18         reinitialise ph;
19     else
20         update ph using the Algorithm 6;
21     update α using the Equation 5.5;
22     i = i + 1;
```

If the current solution produced is at least 85% similar to the best solution (in terms of similarity of their respective paths) then the value of $w1$, the exploration

coefficient, is increased by 1. The similarity function makes use of Levenshtein distance to compute the difference between the current and best path [102]. The 85% similarity is used to account for some degree of variance between paths that are mostly the same without requiring a strict and total match. The algorithm does have a sensitivity to this choice of value. In particular, the higher the value that is chosen, the less likely w_1 is to be increased. The smaller the value, the more likely w_1 is to be increased. As w_1 increases, the influence of the best path is reduced so high values for the similarity comparison enable the best path to remain dominant for longer in the search whereas lower values enable the dominance of the current path. The implication of this is to balance against favouring exploration (low similarity value) or exploitation (high similarity value).

The reason why w_1 is increased, Line 16, is to increase the exploration potential of the algorithm. If the current search is starting to stagnate in terms of its resemblance to the existing path, then increasing $w1$ decreases the influence of the best path by increasing the influence of the current path on the pheromone update process.

The second modification is a pheromone map reset condition. If the current solution is an improvement over the best solution then the 1D pheromone map is reset back to its initial state. This is the mechanism that allows for evaporation of pheromone as it essentially removes the accumulated pheromone when the current solution improves on the existing one, and removing the pheromone allows subsequent paths to explore the space without the added pheromone influencing the search back to the known best path.

5.3 1D Pheromone Updates

The update procedure for a 1D HACO is given in Algorithm 6. For each heuristic under consideration, a count of its frequency in the current ant's path is multiplied against a value w_1. This is then added to a count of the frequency of that heuristic in the best path found multiplied by w_2. These coefficients, w_1 and w_2, balance the exploration/exploitation potential of the algorithm.

Algorithm 6: 1D HACO Update Procedure

Input: A pheromone map ph_C, $w1$, $w2$ the exploration/exploitation coefficients
Result: ph_C or ph_P is updated

```
1 for each heuristic p do
2     δ1=w1*countFrequencyCurr(p);
3     δ2=w2*countFrequencyBest(p);
4     ph(p)+=δ1+δ2;
```

In this case, w_1 weighs the influence of the current solution's heuristics and w_2 weighs the influence of the best heuristic path. Larger values of either would increase the amount of pheromone that is accumulated for that heuristic.

The values of w_1 and w_2 are initially set to 1, with separate coefficients for each ant. This is to ensure that all heuristics have an equal probability of being initially selected during the beginning of the algorithm. During the execution of each ant's iteration procedure, w_1 can be increased to increase exploration.

The 1D HACO does not make use of evaporation because of the pheromone map being reset during the algorithm's execution if the current solution is similar enough to the best-found solution. The act of resetting the map is what provides an evaporation effect. The resetting also provides some degree of exploration as areas of large pheromone concentration are removed and other parts of the map have increased chances of selection again.

Both of these coefficients initially will be set to 1 to have a balanced exploration/exploitation trade-off between exploration and exploitation.

5.4 2D and 3D Pheromone Updates and Evaporation

Once the ants have constructed their path, two updates need to occur in Algorithm 4, (Lines 16–17). Specifically applying the evaporation effect to the pheromone map and then updating the pheromone map with the new values based on the outcome of the fitness evaluations.

The evaporation is based on the following equation:

$$ph_{xyz} = (1 - p) * (ph_{xyz}) \tag{5.1}$$

where x, y refers to the components x and y on layer z. The evaporation process for a 2D pheromone map is identical save for the omission of the z layer.

The update process is the same as the standard Ant System (AS) [11]. The only modification is to the specific pheromone update value, $\Delta\tau^k_{xyz}$. This is given by:

$$\Delta\tau^k_{xyz} = \left\{ \begin{array}{ll} \frac{1}{f(x^k)*len(x^k)} & \text{if link (x,y,z)} \in \text{path } x^k \\ 0 & \text{if link (x,y,z)} \notin \text{path } x^k \end{array} \right\} \tag{5.2}$$

The update procedure for a 2D pheromone map is the same except that the z layer is omitted.

This modification takes the length of the path into account alongside the fitness associated with that path. This gives weight to both parts of the solution, with the incentive being to minimise both the solution quality and the size of the path associated with that fitness.

5.5 Generalised Path Construction

The process for an ant to construct a path does largely depend on the particular type of hyper-heuristic it is being used in. Firstly because specific hyper-heuristics conceptualise and represent the heuristic space in different ways. For

example, selection hyper-heuristics make use of low-level heuristics whereas generation hyper-heuristics make use of components. The second aspect is the size of the path which also depends on the nature of the hyper-heuristic.

Regardless, there is a generalisable process that all ants, regardless of pheromone map and hyper-heuristic, have to follow to build their paths and this is given in Algorithm 7.

Algorithm 7: Path Construction Process

Input: a ant, p_l the limit of the path
Result: P a path of nodes

```
1 P = ∅;
2 while P.size! = p_l do
3 |   Select a node, nd, using the appropriate technique;
4 |   P = P ∩ nd;
```

The process in Algorithm 7 is the skeleton of the process used to decide what to add to the path of the ant currently under consideration. The word node in this context refers to either the components, in the case of a generation hyper-heuristic or low-level heuristics, in the case of a selection hyper-heuristic. As both are represented in the same way on the pheromone map, the general process for building a path is the same. That is, start with an empty path and add nodes to it until the limit has been reached. The limit, p_l depends on the type of hyper-heuristic and this is described in subsequent chapters.

5.6 Desirability Heuristic

The AS upon which the HACO algorithm is based makes use of a desirability heuristic to counterbalance the influence of the pheromone concentration when performing the calculation to decide which node to move to. As the HACO algorithm operates in the heuristic space, and not the solution space, a heuristic needs to be proposed to guide the search process.

The desirability heuristic, h_k, is simply:

$$h_k(x, path) = \frac{1}{count(x, path)} \tag{5.3}$$

where x is the node being considered to add to the path and *path* is the existing path of the ant. The count function returns the number of instances of x in the current path. Hence this desirability heuristic is one of novelty; it will always bias toward the least represented nodes in the path being constructed.

The reason for this particular kind of heuristic is to capitalise on the exploration potential of the algorithm. Novelty in terms of very diverse paths will be favoured by the heuristic but the behaviour of the algorithm is designed such that

it will initially favour the heuristic's influence, leading to more novel and diverse solutions, with a gradual but steady increase in the influence of the pheromone concentration over time, leading to exploitative behaviour that should refine the best solution found thus far. This heuristic is used across all HACO and HACOH algorithms.

5.7 Control Parameters

In terms of the control parameters for the algorithms, they are presented in two sections. The first section covers the variables used for HACO algorithms that use 2D and 3D pheromone maps. The next section is for 1D pheromone maps as they differ in their operation and thus require different variables.

5.7.1 2D and 3D Pheromone Map Variables

There are two control variables used in HACO algorithm: p and α. The former variable is used to control the rate of evaporation during the execution of the algorithm when using a 2D and 3D pheromone map. The latter is used to weigh the desirability of the pheromone value when choosing nodes. During the execution of the algorithm these variables will be modified through the use of a linear change equation which update these values after every iteration t. These are as follows:

$$p_t = (p_{init} - p_{final}) * \frac{it - t}{it} + p_{final} \tag{5.4}$$

where p_{init} and p_{final} refer to the initial and final value of p respectively and t refers to the current iteration and it refers to the maximum number of iterations.

$$\alpha_t = (\alpha_{init} - \alpha_{final}) * \frac{it - t}{it} + \alpha_{final} \tag{5.5}$$

where α_{init} and α_{final} refer to the initial and final value of α respectively.

The values of p and α make use of change functions to adjust their values during the executions of the appropriate algorithms. This decision is based on a wider strategy that is meant to facilitate an optimisation behaviour in the algorithms.

More specifically, the initial and final values of p and α will both be in the range of [0.1,1.0]. A depiction of this curve for 10 iterations, as an example, is given in Figure 5.1.

This gives both these variables a constant and linear change over the length of the algorithm's execution. For α this means that the value will initially be small and increase over time. Since α controls the influence of the pheromone concentration and the influence of the desirability heuristic (Section 5.6) is governed by (1-α), this means that initially, the algorithm will heavily favour the heuristic over the pheromone concentration. Over time it will reverse this position until the pheromone concentration is favoured to the exclusion of the heuristic. In a conventional AS, the heuristic used is meant to offer additional information

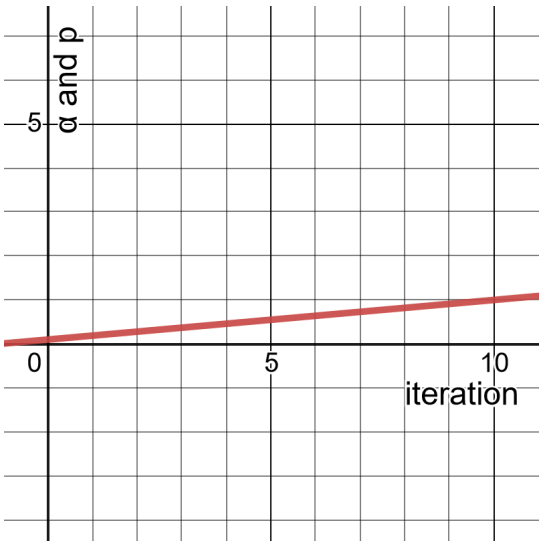

Fig. 5.1: Curve of α and p for 10 Iterations

about the desirability of a node in the context of the solution space. However since this is operating in the heuristic space, a different approach is required. Instead, the search heuristic adds to the explorative potential of the algorithm by favouring the inclusion of novel nodes in the path. The effect of this means that the initial part of the algorithm's execution will explore the heuristic space to a great degree before gradually coalescing around the best paths as decided by pheromone.

The other variable p, which controls the rate of evaporation, is also subject to this same curve and change. As the rate of evaporation is given in Equation 5.1, this means that the initial rate of evaporation will be low with a gradual transition to a high evaporation rate. More specifically, a low value of p means that much more of the pheromone will persist from iteration to iteration. The effect of this will be an initial saturation of pheromone as much of it will persist from iteration to iteration. However as p increases, the rate of evaporation will increase as well which applies a filtering effect on the pheromone as only the strongest, and thus best, regions of pheromone will remain which facilitates exploitation of those remaining regions.

The totality of these two configurations and the behaviour they enable produces an exploration/exploitation trade-off behaviour that provides a general parameter configuration that can be used across a wide array of problems without specific tuning. Hence the reason why this strategy has been adopted in this research.

5.7.2 1D Pheromone Map Variables

HACO algorithms that make use of the 1D pheromone map obviate the need to include the variable p via the update procedure specified in Section 5.3. However, instead of the variable p, it makes use of w_1 and w_2 which are variables that weigh the influence of the best and current paths respectively.

These variables both have the initial value of 1.0 whenever they are used in the experiments. The reason for this is to start the algorithm's execution with an equal bias for both potential influences. Once stagnation starts to occur, that is the best path is similar enough to the current path, w_1 will be increased and allow for more exploration.

5.8 Summary

This chapter has presented the generalised form of the HACO algorithm which later chapters will build on and expand. This includes the high-level overview algorithm and some of the specific details of the approach that are relevant to all of the algorithms regardless of which hyper-heuristic they are. The next chapter will present the specific details of the HACO algorithm as it directly pertains to selection hyper-heuristics.

CHAPTER 6

Ant-Based Selection Hyper-Heuristics

6.1 Introduction

This chapter presents the implementation-specific details that are required for the HACO-SC and HACO-SP algorithms. The previous chapter presented a high-level overview of a generalised HACO algorithm and this chapter provides details that are pertinent to selection constructive and selection perturbative ant-based hyper-heuristics. This chapter covers both of these topics together as selection constructive and selection perturbative hyper-heuristics are not entirely dissimilar in their operation and thus the details presented here are sufficient for both hyper-heuristics. The rest of this chapter is organised as follows. Section 6.2 will discuss how the ant-based selection hyper-heuristics choose the nodes to build their paths. Section 6.3 presents all of the low-level constructive and perturbative heuristics for both of the problem domains. Section 6.4 details the experiments that will be conducted using the algorithms and finally a summary is provided in Section 6.5.

6.2 Node Selection

Algorithm 8 describes the process of choosing nodes for the HACO-SC and HACO-SP algorithms using 3D pheromone maps. The reason for this is that the 1D and 2D pheromone maps follow the same process but just use the smaller dimensional maps instead. This process is applied whenever an ant needs to decide which node to add to its path through the path construction process such as in Algorithm 7. The primary mechanism of node selection is based on the roulette wheel selection via a stochastic acceptance process [103,104]. In terms of the calculation, there are two factors in choosing a node: heuristic desirability, h, and pheromone concentration, ph. The actual selection process makes use of a different formulation of the calculation [105]. By using this formulation, it removes the need for a separate β parameter that is used in the AS.

The variable llh, Line 5, refers to all of the low-level heuristics, constructive or perturbative, that are available for selection. If the limit has been reached then the process will simply return, as a precaution for preventing excessive path construction. The limit, in this case, refers to the size of the path, which for selection hyper-heuristics, is a value dependent on the size of the problem. For selection constructive hyper-heuristics, this is because the path has to stop

Algorithm 8: Node Selection Process

Input: a ant
Result: *choice* the selected node to add to the current path for ant a

1 $set = \emptyset$;
2 **if** $a.currF > limit$ **then**
3 return;
4 **else**
5 $set = llh$;
6 n_{ind}=indexOf(curr node of a.path);
7 l_{ind}=a.path.size()-1;
8 $tmp[] = [set.size()]$;
9 $phs[] = [set.size()]$;
10 $sum_{prob} = 0$;
11 $j = 0$;
12 **for** $j < set.size()$ **do**
13 $tmp[j] = \alpha * ph[n_{ind}][j][l_{ind}] + (1 - \alpha) * h_k(node_j, a.path)$;
14 $sum_{prob} += tmp[j]$;
15 $j = j + 1$;
16 $i = 0$;
17 **for** $i < set.size()$ **do**
18 $pks[i] = \frac{tmp[i]}{sum_{prob}}$;
19 $i = i + 1$;
20 $ind = 0$;
21 $sum = pks[0]$;
22 $r = U(0, 1)$;
23 **while** $sum < r$ **do**
24 $ind = ind + 1$;
25 $sum = sum + pks[ind]$;
26 $choice = set[ind]$;
27 $a.currF += 1$;

at a fully constructed solution and for selection perturbative hyper-heuristics, the size of the problem is a natural limit that allows for a scalable degree of modification to the underlying solution.

To decide on which node to move to, (Lines 11–13), a desirability score has to be calculated that accounts for both pheromone concentration and the novelty heuristic. This movement consists of moving from the current node i to a node j on layer l.

Once the values are calculated for choosing the next node, the roulette wheel with a stochastic acceptance process will take place, Lines 16—24. This process selects the next node (low-level heuristic) to be added to the path. The node with the highest desirability has a proportionally higher chance of selection.

Some considerations need to be made based on the type of pheromone map used in the algorithm. For a 2D pheromone map, the process is identical except that Line 12 would omit the layer index dimension. For the 1D process, the second index, n_{ind}, is removed and only j is needed to iterate through the 1D pheromone map. The skeleton of the procedure remains the same. The only thing that needs modification is how the pheromone is accessed from the map which depends on the type of pheromone map that it is.

6.3 Low-Level Heuristics

This section presents the low-level heuristics considered in both the selection constructive and selection perturbative hyper-heuristics, the HACO-SC and HACO-SP respectively.

6.3.1 Constructive Heuristics

In terms of how the construction heuristics are applied to the problems, an ant's path in the HACO-SC represents a set of heuristics that each contribute something towards the completion of a full solution. This process starts with an initially empty solution and constructs the solution, piece-by-piece, and heuristic-by-heuristic. As a result, the size of an ant's path is typically equal to the number of heuristics required to produce a fully complete solution for that specific problem.

6.3.1.1 QAP Heuristics The construction heuristics used for this problem were developed for this research using the principles discussed in [2]. These heuristics were developed because of the lack of available existing constructive heuristics. What follows is a list of the constructive heuristics where C_i denotes constructive heuristic i:

- C_1: Insert the facility with the most number of flow links to other facilities in the location with the lowest average distance to other locations.
- C_2: Insert the facility with the largest flow value to another facility at a location nearest to where its corresponding facility is.
- C_3: Insert a random facility at the location with the lowest average cost to all other locations.
- C_4: Insert the facility with the smallest flow value to another facility at a location farthest from where its corresponding facility is.
- C_5: Insert the facility with the least number of flow links to other facilities in the location with the highest average distance to other locations.
- C_6: Insert a random facility at a random location.

6.3.1.2 MSSP Heuristics

Given that the MSSP is a new problem, constructive heuristics had to be derived from scratch for this problem. These heuristics were manually derived through the study of the problem domain [78].

What follows is a list of the constructive heuristics where C_i denotes constructive heuristic i:

- C_1: The next scene is randomly chosen.
- C_2: The scene with the most actors is chosen.
- C_3: The scene with the fewest actors is chosen.
- C_4: The scene with the longest duration is chosen.
- C_5: The scene with the shortest duration is chosen.
- C_6: The scene with the smallest transfer cost from the prior scene is chosen.
- C_7: The scene with the largest transfer cost from the prior scene is chosen.
- C_8: A scene is chosen randomly from a list of scenes that share the same location as the prior scene. If no such scenes exist, the next scene is chosen randomly.

6.3.2 Perturbative Heuristics

Perturbative heuristics do not construct a solution from scratch. Rather they perturb an existing solution to improve its quality. The perturbative heuristics will perturb a solution created with the constructive heuristics of the appropriate domain.

6.3.2.1 QAP Heuristics

The perturbative heuristics used for this problem were developed for this research using the principles discussed in [2]. What follows is a list of the perturbative heuristics where P_i denotes perturbative heuristic i:

- P_1: Swap the locations of the facilities with the two highest flow scores. The flow score is defined as $r_1 * r_2$ where r_1, r_2 refer to the flow value between facilities i, j and the locations m, n.
- P_2: Swap the facilities at the best and worst locations (in terms of their average distance to their neighbours) for new random facilities.
- P_3: Perform the 2-opt procedure on the solution permutation.
- P_4: Find a pair of facilities (i, j) with the highest flow score and swap them with the pair of facilities (x, y).
- P_5: Randomly shuffle the solution until an improvement in objective value occurs.
- P_6: Pick two random facilities and swap their locations.
- P_7: Swap two random facilities until an improvement in objective value occurs.
- P_8: Swap the first and last facilities locations.
- P_9: Swap the first-most facility for another random facility and swap the last-most facility for another facility randomly selected.

– P_{10}: Move every facility one location to the right in the solution, with the last facility being moved to the first location.
– P_{11}: Move every facility one location to the left in the solution, with the first facility being moved to the last location.
– P_{12}: Pick a random facility, a pivot point, such that there are at least two facilities to either side of it in the solution and reverse the order of the facilities on either side of the pivot point.

The constructive heuristic used for the initial solution for the QAP domain is a greedy constructive heuristic that inserts the facility with the lowest average cost from the last facility added. This provides a simple heuristic for producing the initial solutions required by the selection perturbative hyper-heuristic.

6.3.2.2 MSSP Heuristics These heuristics were manually derived through the study of the problem domain [78]. What follows is a list of the perturbative heuristics where P_i denotes perturbative heuristic i:

– P_1: The scene order is shuffled until an improvement in fitness occurs.
– P_2: The scene with the longest duration is moved to the front of the schedule.
– P_3: The scene with the shortest duration is moved to the front of the schedule.
– P_4: A random (excluding the end) scene is chosen. From the remaining scenes, the one which has the lowest transfer cost to the original chosen scene is determined. This scene is then put into the adjacent position next to the original random scene.
– P_5: The first and last scenes are interchanged with a corresponding randomly chosen scene from the schedule.
– P_6: The scene with the most number of attached actors is moved to a random position in the schedule.
– P_7: The scene with the least number of attached actors is moved to a random position in the schedule.
– P_8: All of the scenes are shifted up one position in the schedule. The scenes wrap around.
– P_9: All of the scenes are shifted down one position in the schedule. The scenes wrap around.

The constructive heuristic used for the initial solution for the MSSP domain is heuristic C_8 as it was found to have the best performance out of the available construction heuristics [78].

6.4 Experiments and Parameter Choice

The parameter choices for the selection constructive and selection perturbative ant-based hyper-heuristics are detailed in their sections below. The HACO-SC and HACO-SP algorithms will be applied to the MSSP and QAP domains with the same experimental configurations.

6.4.1 Selection Constructive Parameters

Figures 6.1 and 6.2 present the result of the process as described in Section 3.5.3. Pearson's Correlation Coefficient [93] is also calculated for the relationship between fitness and n_k and the number of iterations which are the y and x-axis respectively. A legend is provided at the top of each graph which indicates the number of iterations for each entry.

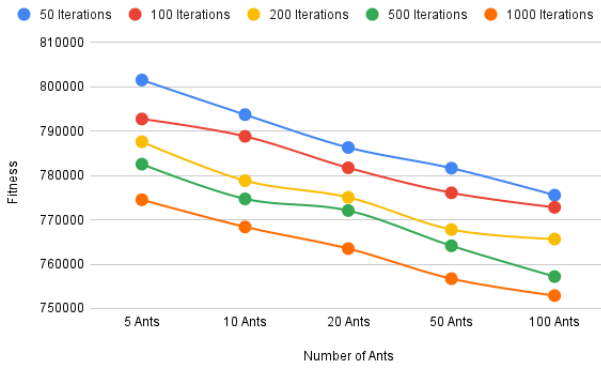

Fig. 6.1: Fitness Comparison between n_k and Number of Iterations

Figure 6.1 shows the difference in fitness values for the HACO-SC algorithm as the number of ants, n_k, and the number of iterations is changed. Correlation coefficients are calculated from this data to determine the strength of the relationship between the variable configurations and the fitness value. The correlation coefficients for the comparison between n_k and fitness, and the number of iterations and fitness, are -0.62 and -0.67 respectively.

In terms of these results, the implications on performance for the HACO-SC are clear. Adding additional ants and iterations do lead to improvements in the overall fitness of the solutions produced by the hyper-heuristic, with a fairly linear relationship between the parameters and the fitness. The correlations are strongly negative, which indicates a strong negative correlation between fitness and the parameters; that is an increase in the parameters strongly correlates to a decrease in fitness.

While there are obvious advantages to using more ants and iterations in the HACO-SC, Figure 6.2 demonstrates that is not without cost. As the parameters increase, the average time to execute a single run increases as well, with the largest values taking several times that of the smaller values. The time taken around the 20 ant mark seems to provide a good enough runtime efficiency in relation to its performance as it performs comparably to other configurations, but with a lower runtime.

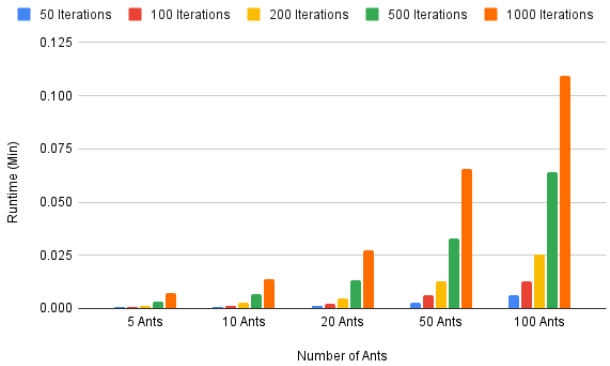

Fig. 6.2: Runtime Comparison between n_k and Number of Iterations

Considering these factors, the choice for n_k and the number of iterations should be a compromise between the increased computational load of the larger values, and the improved performance they offer.

Based on the outcome of these results, the parameters for these experiments are n_k, the number of ants, the number of runs and the number of iterations are:

- n_k: 20
- Number of Runs: 30
- Number of Iterations: 750

These values offer a compromise in terms of n_k as that is the midpoint of the range of ant colony sizes and 750 is between the generally better performing but computationally costly 1000 iterations and the weaker but computationally more efficient 500 iterations.

The parameter of p_l, the path length, is set to the size of the problem. With the selection constructive hyper-heuristic, each heuristic that is selected will build up the solution piece-by-piece. Therefore the length of the path must be as large as the problem to ensure that the solution is fully complete.

6.4.2 Selection Perturbative Parameters

Figures 6.3 and 6.4 present the result of the parameter process as described in Section 3.5.3. Pearson's Correlation Coefficient [93] is also calculated for the relationship between fitness and n_k and the number of iterations respectively. A legend is provided at the top of each graph which indicates the number of iterations for each entry.

Figure 6.3 shows the difference in fitness values for the HACO-SP algorithm as the number of ants, n_k, and the number of iterations is changed. Correlation coefficients are calculated from this data to determine the strength of the

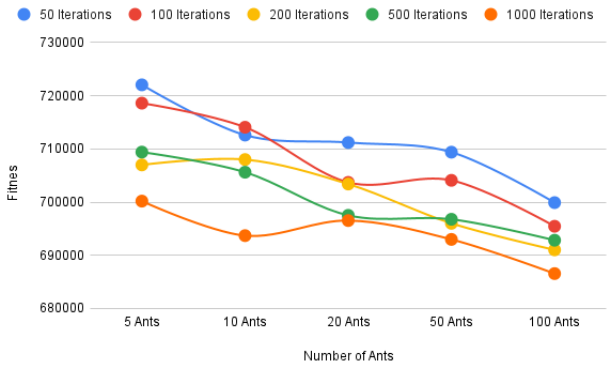

Fig. 6.3: Fitness Comparison between n_k and Number of Iterations

relationship between the variable configurations and the fitness value. The correlation coefficients for the comparison between n_k and fitness, and the number of iterations and fitness, are -0.66 and -0.60 respectively.

The HACO-SP fitness results, Figure 6.3 are similar to the HACO-SC in Figure 6.1 with some differences. Primarily, the distinctions between the different configurations and their fitness results are less strongly differentiated with linear trends being observed but the lines between the configurations are less distinct. In particular, the differences between using 500 iterations and 1000 iterations are reduced going from 10 ants to 20 ants.

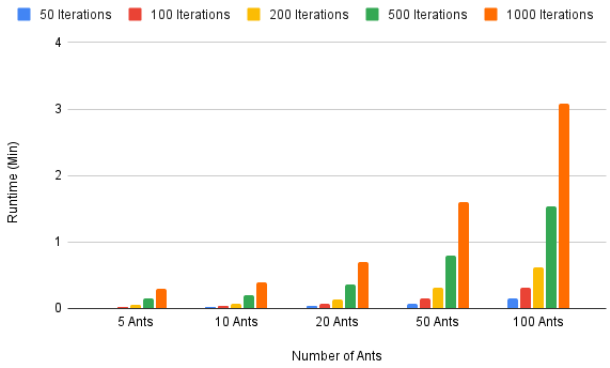

Fig. 6.4: Runtime Comparison between n_k and Number of Iterations

The runtimes, given in Figure 6.4 are consistent with the behaviour seen in Figure 6.4. Larger values of n_k and the number of iterations result in increasingly larger runtimes for a single execution of the HACO-SP algorithm. Of course, the

largest share of this stems from the 1000 iteration configuration which is always the longest-running configuration, regardless of the value of n_k. However, as demonstrated by Figure 6.3, this increased runtime does not necessarily lead to improved performance as the differences can be marginal between the 500 and 1000 iterations, depending on the number of ants.

Based on the outcome of these results, the parameters for these experiments are n_k, the number of ants, the number of runs and the number of iterations are:

- n_k: 20
- Number of Runs: 30
- Number of Iterations: 750

These values offer a compromise in terms of n_k as that is the midpoint of the range of ant colony sizes and 750 is between the generally better performing but computationally costly 1000 iterations and the weaker but computationally more efficient 500 iterations.

The parameter of p_l, the path length, is set to the size of the problem. With the selection perturbative hyper-heuristic, the path length indicates the number of times a low-level perturbative heuristic is applied to a given solution. Setting the path limit to the size of the problem is a simple way of ensuring that the number of permutations that can be applied will always match the complexity of the current problem.

6.5 Summary

This chapter presented relevant implementation details as they pertained to the HACO-SC and HACO-SP. These details included how nodes are selected, the low-level heuristics that are used by the two hyper-heuristics, parameter considerations and how these hyper-heuristics will be used in experiments. The next chapter presents the implementation details of the HACO-GC.

CHAPTER 7

Ant-Based Generation Constructive Hyper-Heuristic

7.1 Introduction

This chapter presents the details of how the HACO algorithm can be used for a generation constructive hyper-heuristic. These adaptations are made to the generalised HACO algorithm. This is a more extensive set of modifications to the baseline algorithm as generation constructive hyper-heuristics differ from their selection hyper-heuristic counterparts in several ways. The rest of this chapter is organised as follows. Section 7.2 presents the modification to the HACO path construction algorithm, Algorithm 7, so that ant paths can be used to construct heuristics. Section 7.4 describes how the created paths can be converted into heuristics. Section 7.3 presents details for the modifications made to the HACO node selection algorithm, Algorithm 7.3. Section 7.5 explains how the constructed heuristics are meant to be interpreted. Section 7.6 describes the problem components in terms of operators and attributes that make up the problem domains. Section 7.7 details how solutions are constructed for the 1BPP and MSSP domains. Section 7.9 details the experiments for the HACO-GC algorithm with a summary of the chapter given in Section 7.10.

7.2 Path Construction

The HACO-GC differs from the HACO-SC and HACO-SP in terms of how it constructs its path in a significant way. The heuristic space of a generation constructive hyper-heuristic consists of components that have to be combined to form a heuristic as opposed to the fully formed heuristics of a selection hyper-heuristic. This difference necessitates a different type of path construction to translate the path taken by the ant into a heuristic that can be interpreted and used.

The process in Algorithm 9 is the skeleton of the process used to convert a set of nodes representing a path into a structured heuristic that represents a control function. The control function is used by the solution construction process to build the solution. This process follows the normal convention for path building, adding nodes one at a time to a path, but simultaneously, a process is applied to structure the path into a heuristic format that can later be interpreted as such for fitness evaluation. Two notational conventions apply to this and subsequent algorithms. Firstly the $\cap$ operator is used to show the addition of sets with

either sets or other elements. Secondly, quotation marks denote the use of string elements.

Algorithm 9: Generation Constructive Path Construction Process

Input: a ant, p evaporation rate
Result: S a heuristic, P a path of nodes

```
1  r = U(0, 1);
2  P = ∅;
3  S = {};
4  a.currF = 0;
5  if r > p or S == ∅ then
6      P = P ∩ random operator from the operator set;
7  else
8      P = P ∩ first node of the best path P_B;
9  a.currF+ = 1;
10 S = "[" ∩ P[0] ∩ ";";
11 art = getArity(P[0]);
12 i = 0;
13 for i < art do
14     c=choose a node using Algorithm 10;
15     P = P ∩ c;
16     comp = compute(ant, c, P, a.currF);
17     S = S ∩ comp;
18     i = i + 1;
19 Remove the last character from S;
20 S = S ∩ "]";
```

The process starts (Lines 1–3) with a blank path and heuristic, P and S respectively and starts by adding an operator node to the path, either from the best path or randomly chosen from the operator set. This choice enables the path construction process to initially make use of the randomly chosen nodes that will help facilitate exploration before gradually moving over to making use of the best path's initial node to guide the search and rely more on the exploitation of prior information.

From that point, it will increase $currF$, the current number of operators for a given path, (Line 9). The process for path construction will terminate when $currF$ is equal to the limit, p_l. The path limit, p_l principally is based on the number of operators allowed in a heuristic. As only operators add additional complexity, by needing inputs to their functions, this is the most important aspect that determines how large a heuristic can grow.

The central mechanism of the algorithm occurs between (Lines 13–18). Here, the additional number of components is based on the arity of the input path.

This function makes use of recursion, (Line 16), which is how it can build nested components inside of others. This process will also resolve the construction of components in a depth-wise manner, expanding any possible nested components (if they are operators) before moving to the next component at the same parent component.

The typical way in which generation constructive hyper-heuristics use the heuristics that they construct is as a control function in a solution construction process. That is the constructed heuristic is used as a part of a wider solution construction strategy with the quality of the heuristic measured in terms of the quality of the solutions it helps to create. Specifically, the heuristic is used to determine some desirability score for parts of a solution during the solution construction process and the solution is built around those calculated scores.

The nature of the algorithm is such that the heuristic returned represents a complete control function and no repair operation will be needed to remedy structural errors. The initial operator that forms the first node in the path will determine, via its arity, the number of additional nodes to add to the path. As more operators are added, those, in turn, will require more nodes to be added until the path is complete. A user-defined limit parameter controls the size of the path as the limit controls how many operators can be added to a path. Finally, it is important to note that when generated, the heuristics make use of a prefix notation with regards to their structure.

7.3 Node Selection

Algorithm 10 describes the process of choosing nodes for a HACO algorithm using a 3D pheromone map. The reason for this is that the 1D and 2D pheromone maps follow the same process but just use the smaller dimensional maps instead. This process is applied whenever an ant needs to decide which node to add to its path through the path construction process. The primary mechanism of node selection is based on the roulette wheel selection via a stochastic acceptance process [103,104]. In terms of the calculation, there are two factors in choosing a node: heuristic desirability, h, and pheromone concentration, ph.

The initial part of the process determines whether the entire set of components (functional and domain attributes) are to be chosen or only the domain attributes (Lines 2–5). Afterwards, a number of necessary initialisations occur (Lines 6–10). The process calculates the pheromone for the current layer at the intersection of the current node i moving to node j at the layer l. This will take place from (Lines 11–13). Once the values are calculated for choosing the next node, the process of selection will take place, (Lines 16—24). The node with the highest desirability has a proportionally higher chance of selection.

For a 2D pheromone map, the process is identical except that (Line 12) would omit the layer index dimension. For a 1D pheromone, the second index, *nind*, is removed and only j is needed to iterate through the 1D pheromone map. The skeleton of the procedure remains the same. The only thing that needs

modification is how the pheromone is accessed from the map which depends on the type of pheromone map that it is.

Algorithm 10: Node Selection Process

Input: a ant
Result: *choice* the selected node to add into the current path for ant a

1 $set = \emptyset$;
2 **if** $a.currF < limit$ **then**
3 $\quad set = domainAttSet \cap operatorSet$;
4 **else**
5 $\quad set = domainAttSet$;
6 n_{ind}=indexOf(curr node of a.path);
7 l_{ind}=a.path.size()-1;
8 $tmp[] = [set.size()], phs[] = [set.size()]$;
9 $sum_{prob} = 0$;
10 $j = 0, i = 0, ind = 0$;
11 **for** $j < set.size()$ **do**
12 $\quad tmp[j] = \alpha * ph[n_{ind}][j][l_{ind}] + (1 - \alpha) * h_k(node_j, a.path)$;
13 $\quad sum_{prob} += tmp[j]$;
14 $\quad j = j + 1$;
15 **for** $i < set.size()$ **do**
16 $\quad pks[i] = \frac{tmp[i]}{sum_{prob}}$;
17 $\quad i = i + 1$;
18 $sum = pks[0]$;
19 $r = U(0, 1)$;
20 **while** $sum < r$ **do**
21 $\quad ind = ind + 1$;
22 $\quad sum = sum + pks[ind]$;
23 $choice = set[ind]$;
24 **if** $choice \in operatorSet$ **then**
25 $\quad a.currF += 1$;

7.4 Heuristic Conversion Process

The underlying ant system traverses through the component space by building a path. However, the path itself requires structuring to be interpreted as a heuristic. This is facilitated by Algorithm 11. This serves as the wrapper for Algorithm 9 which continues the recursive process.

The function revolves around the expression that is passed to it. If the expression is in the domain attribute set, it is returned, (Lines 1–2), with a comma to separate it in the heuristic. Otherwise, the expression represents a function that necessitates choosing more nodes based on the arity of the function. The general structure of the remaining algorithm (Lines 7–13), follows a similar pat-

tern as in the case of Algorithm 9 with the adding and fully expanding any new components in the heuristic.

This conversion process happens as nodes are added to the ant's path. So as the path is added to, its corresponding heuristic is assembled and structured to be interpretable as a control function. Importantly, domain attribute expressions are delimited with commas whereas operator expressions, which can include operators and domain attributes, are delimited with the vertical bars. The heuristic itself is represented as a string expression that represents the combination of domain attributes and operators put into a structured format. Some examples of these expressions are presented below:

```
A,
-:A,B
+:{-:A,C}|{-:A,B}
```

Fig. 7.1: Examples of Expressions

Algorithm 11: Compute Recursive Function

Input: a ant, exp a expression representing a component, v a set of variables about the problem state, $currF$ the current number of operators in the path

Result: res a heuristic expression

```
1  if exp ∈ domainAttSet then
2      return exp ∩ ",";
3  else
4      res = "{" ∩ exp ∩ " : ";
5      art = getArity(exp);
6      i = 0;
7      for i < art do
8          c=choose a node using Algorithm 10;
9          P = P ∩ c;
10         res = res ∩ compute(ant, c, P, a.currF);
11         i = i + 1;
12     remove the last character from res;
13     res = res ∩ "}|";
14     return res;
```

The compute function, Algorithm 11, does the conversion of the path node into the string heuristic representation. The process is largely the same as the skeleton function, Algorithm 9, but with the addition of the return exp+ state-

ment, Line 2. This statement will return the domain attribute to add to the heuristic but does not entail additional additions to the path as it does not return an operator like the other return statement on Line 12. By contrast, an operator requires inputs based on its arity value. These inputs necessitate adding new components to the path.

7.5 Path Interpretation

In terms of path interpretation, the process functions in reverse to the construction process. The heuristic will be interpreted recursively from the outermost operator element to the innermost nested element. The interpretation of the heuristic function converts the heuristic into an equation where the domain attributes are replaced with their corresponding values where required and then processed as their inputs to the operator inputs which then returns the final value to be used in the solution construction process.

Consider a path represented below:

```
+ -> A -> * -> A -> B
```

which would then be converted into the following heuristic:

```
[+:A,{*:A,B}]
```

which would then be interpreted as the equation:

```
A+(A*B)
```

This construction process follows a depth-wise process, with a node being fully expanded (in terms of the recursive process) before adding the next choice in the function inputs.

7.6 Problem Components

This section describes the low-level components that are used for the problems. These are namely operators and domain attributes. The domain attributes are specific to each problem domain whereas the operators are universal across all domains.

7.6.1 Operators

The 1BPP and MSSP domain both use the same set of operators. These operators define a range of possible arithmetic operations that can be done in the context of a generated heuristic. The operators are given by Table 7.1.

The division function is protected from returning undefined values if the denominator is zero. If the denominator is zero, it will simply return the numerator instead. The A operator refers to the absolute value. The arity column indicates the number of inputs required for that function. An operator with an arity of 2 for instance requires two inputs.

Table 7.1: Domain Operators

Element	Arity
+	2
-	2
*	2
/	2
A	1

7.6.2 Domain Attributes

The attributes used for the 1BPP and MSSP domains are given below. These attributes are used in conjunction with the operators to build heuristics.

7.6.2.1 1BPP Attributes In terms of the 1BPP, the domain attributes are taken from [106]. In their work, the authors provided three domain attributes which are described below:

- F: returns the sum of the pieces already in the bin.
- C: returns the bin capacity.
- S: returns the size of the current piece.

These are simple domain attributes that reflect the state of the packing process.

7.6.2.2 MSSP Attributes The components were taken from [78] to reflect basic characteristics of the problem state during the construction process and they are described below:

- d: Duration of a given scene.
- a: average wage of the actors attached to a given scene.
- lp: number of already scheduled scenes that share a location with the current scene.
- tc: one divided by the transfer cost from the last scheduled scene to the current given scene. Returns 1 if there are no scheduled scenes.
- td: one divided by the transfer time from the last scheduled scene to the current given scene. Returns 1 if there are no scheduled scenes.
- an: number of actors assigned to the current scene.

7.7 Solution Construction Process

Typically a generation constructive hyper-heuristic will evolve a control function. A control function is a construction heuristic that guides the construction process as it constructs a solution for a given problem. This control function calculates a desirability score used to determine which parts of the solution to add during

construction. The term control function represents the fact that the construction heuristic is used to control the solution construction process.

For different problems, the desirability score represents different aspects of the problem, like the desirability to add a given node to a current path for example. Each of the algorithms presented below make use of the evolved heuristics as their control functions which indicate which parts of the solution should be added during their respective solution construction process.

7.7.1 1BPP

Algorithm 12: 1BPP Construction Method

Input: S a heuristic, *items* a set of items to pack
Result: *sol* a constructed solution

```
1  sol+ =new empty bin;
2  scores = [];
3  while items ≠ ∅ do
4      scores=[items.size()];
5      i = 0;
6      for i < items.size() do
7          scores[i]=evaluateHeuristic(S,items[i]);
8          i = i + 1;
9      choice = max(scores);
10     if sol.currBin is full then
11         sol = sol ∩ new empty bin;
12     else
13         sol.currBin = sol.currBin ∩ items.remove(choice);
```

Algorithm 12 describes solution construction process for the 1BPP domain. This algorithm will construct a complete solution. Starting with a given heuristic and a set of items to pack, the process starts with a single bin. The current bin being packed is referred to as *currBin*. All of the remaining items are evaluated based on the heuristic with the highest scoring item, Line 7, chosen to be added to the current bin.

The *scores* variable, Line 2, is an array that holds the desirability score determined by each heuristic S that is calculated by the *evaluateHeuristic* function. The MSSP solution construction processes will make use of the *scores* variable in the same way.

The score represents the desirability of choosing a given item to add to the bin based on its current state and the problem overall.

If the item cannot fit, a new bin is opened and the process is repeated until all items are packed. Rather than making assumptions about the number of bins, this process will arrive at the number of bins through the packing process. The value of the heuristic determines how much space is wasted in each bin with better heuristics minimising wasted space and thus using fewer bins. This

strategy is drawn from existing literature where the solution construction process makes item selections as opposed to bin selections [2].

The reasoning behind this approach is to arrive at the number of bins for the solution organically. As all of the bins are of the same capacity, the task of packing an arbitrary number of items into all of the bins could be reduced down to packing them in a single bin. Hence the focus is shifted towards developing a heuristic that minimises wasted space in a bin with the final number of bins reflecting the degree to which this function was successful. A more optimal heuristic will need several bins closer to the actual optimal number of bins without having to specify the number of bins beforehand or apply additional repair methods to a constructed solution.

7.7.2 MSSP

Algorithm 13: MSSP Construction Method

Input: S a heuristic, *scenes* a set of scenes
Result: *sol* a constructed solution

```
1  sol = ∅;
2  scores = [];
3  while scenes ≠ ∅ do
4      scores=[scenes.size()];
5      i = 0;
6      for i < scenes.size() do
7          scores[i]=evaluateHeuristic(S,scenes[i]);
8          i = i + 1;
9      choice = max(scores);
10     sol = sol ∩ scenes.remove(choice);
```

Algorithm 13 describes the process by which a given MSSP solution is constructed. The given scenes need only be added to a vector to form a permutation. The score represents the desirability of adding a given scene to the solution as the next scene to be scheduled. The process uses the maximum score, Line 7, due to the choice of the domain attributes.

7.8 Comparison Heuristics

Several existing heuristics will be used for comparison against the HACO-GC in the 1BPP domain. These heuristics were taken from literature and are widely used packing heuristics [2]. These heuristics are construction heuristics that build solutions incrementally.

The construction heuristics are as follows:

- First Fit (FF)
- Best Fit (BF)
- Next Fit (NF)

– Worst Fit (WF)
– First Fit Decreasing (FFD)
– Best Fit Decreasing (BFD)
– Next Fit Decreasing (NFD)

7.9 Experiments and Parameter Tuning

Figures 7.2 and 7.3 present the result of the process as described in Section 3.5.3. Pearson's Correlation Coefficient [93] is also calculated for the relationship between fitness and n_k and the number of iterations respectively. A legend is provided at the top of each graph which indicates the number of iterations for each entry.

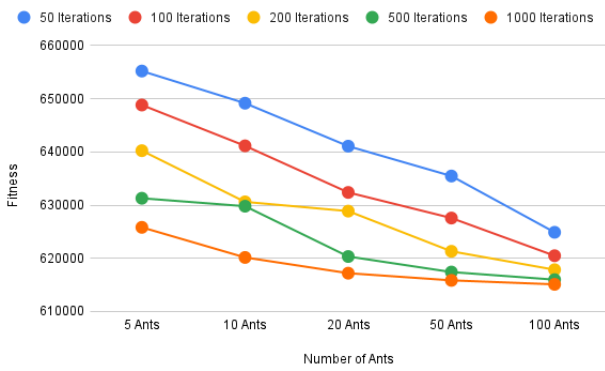

Fig. 7.2: Fitness Comparison between n_k and Number of Iterations

Figure 7.2 shows the difference in fitness values for the HACO-GC algorithm as the number of ants, n_k, and the number of iterations is changed. Correlation coefficients are calculated from this data to determine the strength of the relationship between the variable configurations and the fitness value. The correlation coefficients for the comparison between n_k and fitness, and the number of iterations and fitness, are -0.60 and -0.61 respectively.

Based on the results shown in Figure 7.2 there is more of a clear delineation between the number of iterations and n_k and the effect they have on produced fitness values. In particular, the strong linear trend returns, with strong negative correlations with the number of iterations, n_k and the fitness values. There is also a flattening of this trend in the higher iteration values (500 and 1000), especially as the number of ants increases from 20.

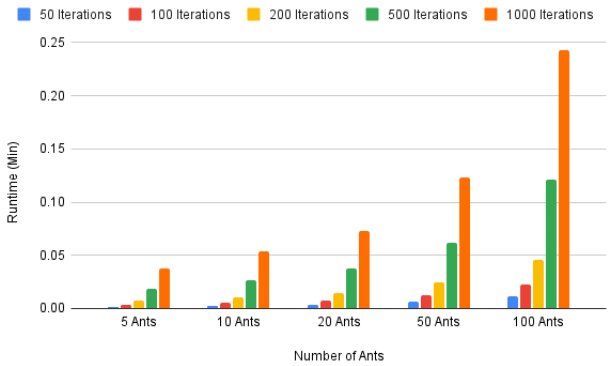

Fig. 7.3: Runtime Comparison between n_k and Number of Iterations

The runtime results in Figure 7.3 indicate the same trends as in the prior runtime comparisons (Figures 6.2 and 6.4). The major factor influencing the runtime is the number of iterations with the number of ants contributing, but not as severely to the increased runtimes. Again, the 20 ant point shows a comparatively low runtime despite a very comparable performance in fitness values as compared to the large configurations.

Based on the outcome of these results, the parameters for these experiments are n_k, the number of ants, the number of runs and the number of iterations are:

- n_k: 20
- Number of Runs: 30
- p_l: 10
- Number of Iterations: 750

The size of the path, p_l, determines the size of the construction heuristic that can be created, in terms of the number of operators that the construction heuristic can contain. During the development of the HACO-GC, the algorithm rarely produced heuristics with more operators than 10, even when given larger limits. The built-in incentive to produce heuristics of minimal size contributes to this and thus the chosen limit represents a good value for allowing for large heuristics to be made whilst still allowing for smaller heuristics to be preferred.

These values offer a compromise in terms of n_k as that is the midpoint of the range of ant colony sizes and 750 is between the generally better performing but computationally costly 1000 iterations and the weaker but computationally more efficient 500 iterations. The HACO-GC algorithm will be applied to the 1BPP and MSSP domains.

7.10 Summary

This chapter presented implementation details specific to the HACO-GC algorithm. This includes the necessary modifications that explain how the algorithm

can generate new heuristics. The next chapter will present the HACO-GP approach which describes how the HACO algorithm is applied for generation perturbative hyper-heuristics.

CHAPTER 8

Ant-Based Generation Perturbative Hyper-Heuristic

8.1 Introduction

This chapter presents the HACO-GP algorithm, the ant-based generation perturbative hyper-heuristic. This is the most novel application of the HACO methodology as generation perturbative hyper-heuristics are a relatively understudied area of hyper-heuristics. The HACO-GP algorithm is explained in detail including what kind of perturbative heuristics it is capable of generating. The rest of this chapter is organised as follows. Section 8.2 explains how perturbative heuristics can be generated from components and what those components would be. Section 8.3 details the necessary changes to the path construction process for the HACO-GP. Section 8.4 presents the process to convert an ant's path into a perturbative heuristic. Section 8.3.1 details the changes made to the node selection process for the HACO-GP. Section 8.5 presents all of the components for the CVRP and MSSP problem domains considered for this algorithm. Section 8.7 explains the experiments that will be conducted using the HACO-GP algorithm. Finally, Section 8.8 provides a summary of the chapter.

8.2 Generating Perturbative Heuristics from Components

Before the HACO-GP algorithm can be explained in detail, it is necessary to first explain how perturbative heuristics will be generated and from what elements. The general model of the perturbative heuristic will follow a similar structure to those generated by the HACO-GC algorithm in the sense that they will be comprised of two types of components that are combined into a single perturbative heuristic that can be applied to an initial solution.

8.2.1 Selectors and Mutators

The first type of component is called the selector. It is a loose analogue for the domain attributes used by the HACO-GC. Like a domain attribute, the selector is a component that has a descriptive value of the given problem at hand, but unlike the attribute, the selector is specifically so-called because it makes some kind of selective decision regarding some aspect of the problem being considered.

More specifically, selectors represent decision functions that are capable of deterministically selecting aspects of a given problem to be used as inputs by

the mutators which are the second type of component. For example, a selector could decide which route from a list of routes should be given to a mutator for modification.

Mutators are so-called because they represent move operators that are capable of modifying a given solution in some way. They are a loose analogue for the operators in the HACO-GC in the sense that they receive inputs from the selectors and then perform an operation based on those inputs.

Consider a basic move operator such as

```
swap(X1,X2)
```

In this case, the swap function itself serves as the mutator which will swap two elements, X1 and X2, that are chosen by selectors.

Through the combination of these two elements, it is, therefore, possible to create extended heuristics that apply multiple operations to a given problem to produce the best possible refinement.

A basic example of such a composition would be

```
swap(swap(X1,X2),swap(X3,X4))
```

where several swap operations are themselves nested inside a swap operation with multiple selectors provided to select which elements in the problem should be considered for the swap operation. For example, for a basic routing problem, the selectors might select from the available vertices that make up the problem.

8.2.2 State-Based Transition

The model of perturbation that is employed by these heuristics is a state-based transition model. This means that a generated heuristic starts with an initial solution state for the problem being solved.

It then linearly interprets the heuristic, from the first mutator. These mutators modify the initial solution in some way before passing the new state to the next mutator or selector for its operation. The single state is updated as the operation proceeds so that the solution state is always updated to reflect the most recent changes.

This process must be as deterministic as possible as randomness in the outcomes of the operations can drastically affect the quality of the feedback relating to the effect of the heuristics. If the same heuristic can be applied but have a different effect because some of its components are stochastic, it creates a challenge for assessing the real capacity of the heuristic.

To understand the state transition process, consider a simple example

```
swap(swap(X1,X2),swap(X3,X4))
```

This heuristic can be represented in terms of a state transition diagram which is given in Figure 8.1

In this diagram, some nodes represent the mutator, swap, and selectors represented by X1–X4. The diagram indicates the transition of the states over the

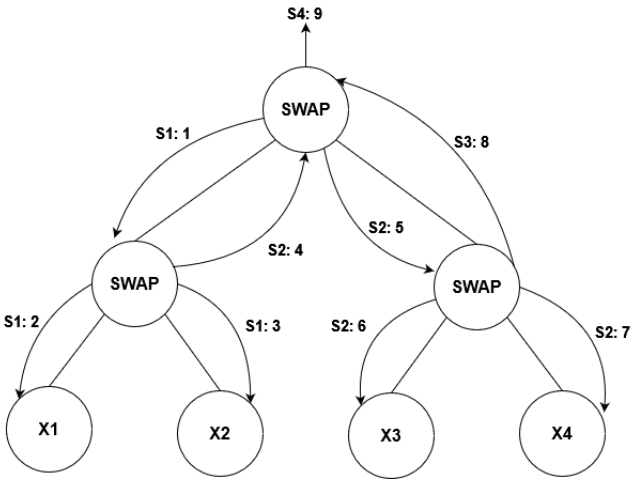

Fig. 8.1: State Transition Diagram

interpretation of the heuristic. In the diagram, the arcs represent states and the transitions between them. The nodes represent the actions that are used to modify the states.

From S1, which represents the initial solution state, this is given to the left-most swap which passes it to the selectors it has as arguments, X1 and X2. Once those selectors decide on the selection the swap mutator will make the swap, transitioning the state from S1 to S2 passing that back to the prior swap. The process repeats for the second swap before the final swap.

In total, the three mutators make three state transitions of the problem from its initial state to the final state after the execution of the heuristic. Larger heuristics, that incorporate more elements, will perturb the solution more as they incorporate more mutators.

In terms of the implementation, the mutators will modify a shared collective problem state that is passed through the heuristic to both mutators and selectors as it becomes updated. Only a version of the state exists at any one time to prevent issues with a deadlock or state incoherence.

The selector itself returns its selection made based on the problem state that it receives. The exact nature of the selection will depend on the nature of the problem.

8.2.3 Component Nesting

An important part of the operation of the mutators and selectors is the ability to nest components within each other. In the HACO-GC algorithm, an operator can nest other operators within it by having those operators as its arguments. This works because the operator is capable of returning the result of its operation.

A plus nested inside of another plus can return its sum to be added to the parent plus, for instance. However this is slightly more complicated in the case of the HACO-GP as it is based on the transition of states, and not purely calculations.

The way this is handled in the HACO-GP is to add an extra argument to all mutators which is called the return selector. More specifically, every mutator will have a list of arguments that it accepts. These arguments are selectors that indicate what components need to be considered by the selector.

To create complex heuristics, mutators must be able to be nested inside other mutators. More specifically, a mutator can be an argument to another mutator. However, since mutators do not, on their own, return anything a modification is required to ensure mutators can be nested.

Each mutator defines one additional argument that defines what that mutator returns so any parent mutator knows what selection it is going to be using in its mutation process. This additional argument is called a return selector and it is a selector that returns a component based on the modified state of the mutator operation.

Using this information, an example of a perturbative heuristic constructed in this way could look like

```
[M8:{M9:S10,S7}|S16]
```

where MX refers to mutator X and SX refers to selector X. In this way, it is now possible to construct perturbative heuristics with a similar methodology to the constructive heuristics with the same generation methodology.

8.3 Path Construction

The HACO-GP differs from the HACO-GP in several ways. The most similar part is the path construction process which is given in Algorithm 14. Like the HACO-GC, the HACO-GP has to combine several components to make a heuristic but unlike the HACO-GC there are an additional number of considerations that have to be considered for the process to achieve the desired effect. The process for building a path for the HACO-GP is given by Algorithm 14. It is largely the same as the HACO-GC process, Algorithm 9 with one major difference.

This major deviation takes place from Line 13. Specifically, accommodation is made to the path construction process to account for the requirement of the return selector needed for mutators. The if statement at Line 13 checks if the given argument being added to the path is the last one for that mutator and specifically chooses a selector as opposed to choosing normally. The rest of the algorithm proceeds as it does in Algorithm 9.

Algorithm 14: Generation Perturbative Path Construction Process

Input: a ant, p evaporation rate
Result: S a heuristic, P a path of nodes

```
1  r = U(0, 1);
2  P = ∅;
3  S = ;
4  a.currF = 0;
5  if r > p or S == ∅ then
6      P = P ∩ random operator from the operator set;
7  else
8      P = P ∩ first node of the best path P_B;
9  a.currF+ = 1;
10 S = "[" ∩ P[0] ∩ ";";
11 art = getArity(P[0]);
12 i = 0;
13 for i < art do
14     if i==art-1 then
15         c=choose a selector using Algorithm 10
16     else
17         c=choose a node using Algorithm 10;
18     P = P ∩ c;
19     comp = compute(ant, c, P, a.currF);
20     S = S ∩ comp;
21     i = i + 1;
22 Remove the last character from S;
23 S = S ∩ "]";
```

8.3.1 Node Selection

The node selection process is the same as the HACO-GC which is given in Algorithm 10. The only modification is a provision for limiting the scope of the node selection to choose from the selectors only when choosing the return selector for a mutator as indicated in Algorithms 14 and 15.

8.4 Heuristic Conversion Process

A recursive process is used to produce the perturbative heuristic. This process works by converting an ant's path into its heuristic form. This is similar to the process used by the HACO-GC in Algorithm 11. The HACO-GP version is given in Algorithm 15. The major deviation of this algorithm is given from Line 7. Specifically, the algorithm also applies the additional condition to ensure that the appropriate return selector is chosen during the recursive conversion process.

Algorithm 15: Compute Recursive Function

Input: a ant, exp a expression representing a component, v a set of variables about the problem state, $currF$ the current number of operators in the path

Result: res a heuristic component

```
1  if exp ∈ selectorSet then
2      return exp ∩ ",";
3  else
4      res = "{" ∩ exp ∩ " : ";
5      art = getArity(exp);
6      i = 0;
7      for i < art do
8          if i==art-1 then
9              c=choose a selector using Algorithm 10
10         else
11             c=choose a node using Algorithm 10;
12             i = i + 1;
13         P = P ∩ c;
14         res = res ∩ compute(ant, c, P, a.currF);
15     remove the last character from res;
16     res = res ∩ "}|";
17     return res;
```

8.4.1 Path Interpretation

Once a path has been constructed the interpretation process relies on the state-based interpretation process outlined in Section 8.2.2. An initial solution is provided to the given heuristic. This initial solution is constructed with construction heuristics. Then the heuristic will be parsed component by component, mutator by mutator until each state transition has been applied and the final state of the solution has been reached.

8.5 Domain Components

This section describes the components, selectors and mutators, used for the HACO-GP in both the CVRP and MSSP domains. These components were produced by decomposing existing heuristics into both mutators and selectors.

8.5.1 Selectors

In this section, the selectors used for each of the problem domains are defined. SX refers to selector X.

8.5.1.1 CVRP

The CVRP selectors are as follows.

1. S1: Select the vertex with the highest demand.
2. S2: Select the vertex with the lowest demand excluding the depot.
3. S3: Select the vertex which is the midpoint of a list of all the vertices sorted by demand.
4. S4: Select the next vertex from a persistent counter that will start at the first non-depot vertex and increment to the next one each time this selector is called. This will continuously cycle through all of the vertices.
5. S5: Select the first non-depot vertex.
6. S6: Select the last vertex.
7. S7: Select the midpoint vertex from an ordered list of all of the vertices.
8. S8: Select the most costly vertex. Cost is defined as the cost to move from this vertex to the next plus the cost to move to it from another prior vertex. This can be considered as $A - X - B$ where the cost of vertex X is the cost to move from A to X added to the cost to move from X to B.
9. S9: Select the least costly vertex. Cost is defined as the cost to move from this vertex to the next plus the cost to move to it from another prior vertex. This can be considered as $a - x - b$ where the cost of vertex x is the cost to move from a to x added to the cost to move from x to b.
10. S10: Select the midpoint vertex taken from a list of all vertices sorted by cost. Cost is defined as the cost to move from this vertex to the next plus the cost to move to it from another prior vertex. This can be considered as $a - x - b$ where the cost of vertex x is the cost to move from a to x added to the cost to move from x to b.
11. S11: Select the vertex that is farthest from the depot.
12. S12: Select the non-depot vertex that is closest to the depot.
13. S13: Select the vertex which has the largest depot loop value. A depot loop is a cost to move from the depot to the vertex and back to the depot again.
14. S14: Select the vertex which has the smallest depot loop value. A depot loop is a cost to move from the depot to the vertex and back to the depot again.
15. S15: Select the vertex which has been selected most frequently in the perturbation process. A running total of the frequency of each vertex's selection should be maintained while the heuristic is being interpreted.
16. S16: Select the vertex which has been selected least frequently in the perturbation process. A running total of the frequency of each vertex's selection should be maintained while the heuristic is being interpreted.
17. S17: Select the midpoint vertex which has been selected from a list of all vertices sorted by their frequency of selection. A running total of the frequency of each vertex's selection should be maintained while the heuristic is being interpreted.

8.5.1.2 MSSP

The MSSP selectors are as follows.

1. S1: Select the scene with the most number of actors associated with it.
2. S2: Select the scene with the least number of actors associated with it.

3. S3: Select the midpoint scene from a list of all scenes sorted by the number of actors associated with it.
4. S4: Select the midpoint scene from a list of all scenes sorted by the scene cost. The scene cost is given as the cost of paying all of the actors for that scene plus the cost of transitioning to and from that scene in the scheduling order.
5. S5: Select the listed first scene.
6. S6: Select the last scene.
7. S7: Select the midpoint scene from a list of all scenes.
8. S8: Select the scene with the highest scene cost. The scene cost is given as the cost of paying all of the actors for that scene plus the cost of transitioning to and from that scene in the scheduling order.
9. S9: Select the scene with the lowest scene cost. The scene cost is given as the cost of paying all of the actors for that scene plus the cost of transitioning to and from that scene in the scheduling order.
10. S10: Select the scene with the highest filming duration.
11. S11: Select the scene with the lowest filming duration.
12. S12: Select the midpoint scene from a list of all scenes sorted by scene duration.
13. S13: Select the scene with the highest transfer time from its current location in the scheduling order.
14. S14: Select the scene with the lowest transfer time from its current location in the scheduling order.
15. S15: Select the midpoint scene from a list of all scenes sorted by the transfer time from its current location in the scheduling order.
16. S16: Select the next scene from a persistent counter that will start at the first scene and increment to the next one each time this selector is called. This will continuously cycle through all of the scenes.
17. S17: Select the scene which has been selected most frequently in the perturbation process. A running total of the frequency of each scene's selection should be maintained while the heuristic is being interpreted.
18. S18: Select the scene which has been selected least frequently in the perturbation process. A running total of the frequency of each scene's selection should be maintained while the heuristic is being interpreted.
19. S19: Select the midpoint scene which has been selected from a list of all scenes sorted by their frequency of selection. A running total of the frequency of each scene's selection should be maintained while the heuristic is being interpreted.

8.5.2 Mutators

In this section, the mutators used for each of the problem domains are defined. MX refers to mutator X. The inputs, in terms of selectors, are given in round brackets next to the selector. For example, M1(A, B) is the first selector that takes two arguments which are selectors A and B; any of the selectors can be used as inputs.

8.5.2.1 CVRP

The CVRP mutators are as follows.

1. M1(A, B): Swap two vertices defined by selector A and B.
2. M2(A): Reverse the order of the route that contains the vertex selected by selector A.
3. M3(A): Shuffle all vertices one position to the right in the route that contains the vertex selected by selector A.
4. M4(A): Shuffle all vertices one position to the left in the route that contains the vertex selected by selector A.
5. M5(A): Swap the vertex selected by selector A with the lowest cost swap with any other vertex in its existing route. The cost is defined by the cost of the route.
6. M6(A): Insert the vertex selected by selector A with the lowest cost insertion with any other vertex in its existing route. The cost is defined by the cost of the route.
7. M7(A): Sort the route that contains a vertex selected by selector A in ascending order of demand.
8. M8(A): Sort the route that contains a vertex selected by selector A in descending order of demand.
9. M9(A): Send the vertex selected by selector A to the front of the route that it is found in.
10. M10(A): Send the vertex selected by selector A to the back of the route that it is found in.
11. M11(A, B, C, D): An if-then-else operation that will compare the vertex cost of two vertices selected by selector A and B. If the first vertex's cost is less than or equal to the second's cost, then evaluate and return the third argument C otherwise evaluate and return the fourth argument D as the return selector.
    This can be considered as $a - x - b$ where the cost of vertex x is the cost to move from a to x added to the cost to move from x to b.
12. M12(A, B, C, D): An if-then-else operation that will compare the vertex cost of two vertices selected by selector A and B. If the first vertex's cost is greater than the second's cost, then evaluate and return the third argument C otherwise evaluate and return the fourth argument D as the return selector.
    This can be considered as $a - x - b$ where the cost of vertex x is the cost to move from a to x added to the cost to move from x to b.
13. M13(A): Remove the vertex selected by selector A and find the lowest cost insertion point in all routes. The cost is defined by the cost of the route.

All of the mutators are protected in the sense that they will not consider the depot vertices when making the mutations. A route's integrity in terms of the depot positions will always be maintained by the operations. Additionally, no mutator will violate any of the CVRP constraints through its operation.

8.5.2.2 MSSP

The MSSP mutators are as follows.

1. M1(A, B): Swap the two scenes selected by selectors A and B.

2. M2(A, B, C): Interchange the positions of the scenes selected by selectors A, B and C.
3. M3(A, B): Reverse the scene order of scenes from the position of the scene selected by selector A to selector B.
4. M4(A, B, C, D): An if-then-else operation that will compare the scene cost of two scenes selected by selectors A and B. If the first scene's cost is less than or equal to the second's cost, then evaluate and return the third argument C otherwise evaluate and return the fourth argument D as the return selector. The scene cost is given as the cost of paying all of the actors for that scene plus the cost of transitioning to and from that scene in the scheduling order.
5. M5(A, B, C, D): An if-then-else operation that will compare the scene cost of two scenes selected by selectors A and B. If the first scene's cost is greater than the second's cost, then evaluate and return the third argument C otherwise evaluate and return the fourth argument D as the return selector. The scene cost is given as the cost of paying all of the actors for that scene plus the cost of transitioning to and from that scene in the scheduling order.
6. M6(A, B): Create a hash list of all of the scenes between the scene selected by selector A and B. Sort this list in ascending order and then transplant all the scenes in this new order back into their original positions. The hash sorting guarantees that the list will be sorted deterministically.
7. M7(A, B): Create a hash list of all of the scenes between the scene selected by selector A and B. Sort this list in descending order and then transplant all the scenes in this new order back into their original positions. The hash sorting guarantees that the list will be sorted deterministically.
8. M8(A): Send the scene selected by selector A to the front of the scene order.
9. M9(A): Send the scene selected by selector A to the back of the scene order.
10. M10(A): Swap the scene selected by selector A with the scene in the scene order that results in the best improvement in solution quality.

All of the mutators are protected in the sense that the integrity of the schedule cannot be removed through any of the mutator operations.

8.5.3 Application of the Heuristic

The application of the heuristic is a simple one. An initial solution will be generated, based on the same guidelines as the HACO-SP, Section 6.3.2. The CVRP makes use of the Clarke-Wright Saving algorithm as the construction heuristic to produce the initial solution [107]. The generated heuristic will then be applied to the initial solution in its totality and the fitness of the solution after the process will be assessed.

8.6 Comparison Heuristics

A number of perturbative heuristics are needed for comparison against the HACO-GP in the CVRP domain. These heuristics are taken from existing literature [45]. The perturbative heuristics are as follows:

– P_1: Select one node randomly and move it into any random route.
– P_2: Select two random nodes and swap their routes.
– P_3: Select a random route and reverse a tour between two nodes.
– P_4: Select three random nodes and exchange their routes randomly.
– P_5: Perform the 2-opt procedure on a random route.
– P_6: Do the 2-opt procedure to all routes.
– P_7: Select two random distinct routes and swap the first portion of the route with the first portion of the second route.
– P_8: Select two distinct random routes and swap the adjacent node of a randomly selected node in each route.
– P_9: Select two random distinct routes and swap the first portion of the first route with the end portion of the last route.
– P_{10}: Pick a random route and move a randomly selected node in that route to a new random position in that same route.

The comparison heuristics for the MSSP domain are taken from Section 6.3.2.

8.7 Experiments and Parameter Tuning

Figures 8.2 and 8.3 present the result of the process as described in Section 3.5.3. Pearson's Correlation Coefficient [93] is also calculated for the relationship between fitness and n_k and the number of iterations respectively. A legend is provided at the top of each graph which indicates the number of iterations for each entry.

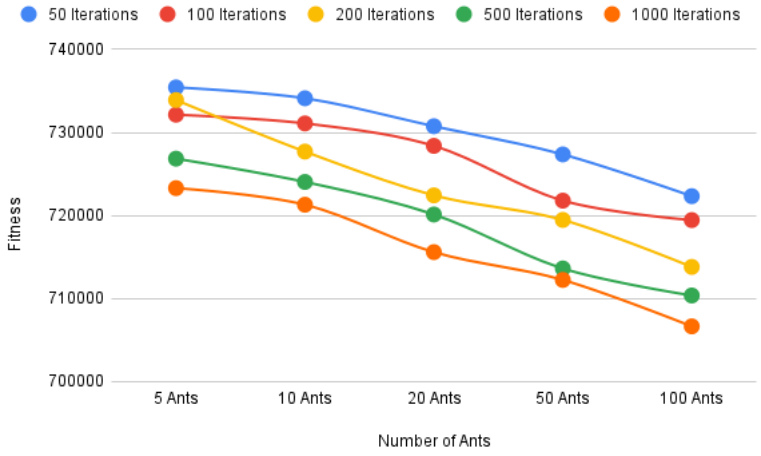

Fig. 8.2: Fitness Comparison between n_k and Number of Iterations

Figure 8.2 shows the difference in fitness values for the HACO-GP algorithm as the number of ants, n_k, and the number of iterations are changed. Correlation coefficients are calculated from this data to determine the strength of the relationship between the variable configurations and the fitness value. The correlation coefficients for the comparison between n_k and fitness, and the number of iterations and fitness, are -0.70 and -0.61 respectively.

The results, Figure 8.2 show strong negative correlations between the parameters and the fitness value of the HACO-GP algorithm. These results are consistent with the HACO-GC results with the higher-valued configurations doing better than lower ones in general, although again, the margins between the 500 and 1000 iterations are quite narrow.

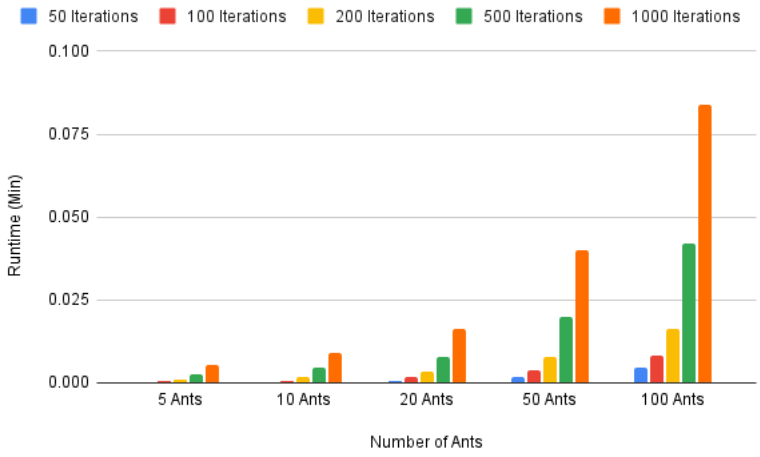

Fig. 8.3: Runtime Comparison between n_k and Number of Iterations

The runtimes shown in Figure 8.3, continue the trends observed in the prior Figures 6.2,6.4 and 7.3. Namely the 1000 iteration configuration produces the highest runtime values for any configuration of ants. Given the relative computational costs, and the fact that Figure 8.2 indicates that the performance between the 500 and 1000 iteration configuration is not that large, opting for the less computationally intensive configuration is preferable.

Based on the outcome of these results, the parameters for these experiments are n_k, the number of ants, the number of runs and the number of iterations are:

- n_k: 20
- Number of Runs: 30
- p_l: 50
- Number of Iterations: 750

The size of the path, p_l, determines the number of perturbative operations, mutators, that are allowed in a single heuristic. In theory, more operations would result in more chances to improve the solution, but in practice, during development, it was found that a limit needs to be imposed to prevent excessive replication of components that do little to improve the actual quality of the heuristics. During the development of the HACO-GP algorithm, testing found that increasing the path limit would result in improved fitness results but also increase the computational costs of the algorithm. The limit of 50 provided the best performance for a reasonable computational cost.

The experiments here are meant to assess the overall methodology and a limit of 50 is used as a value large enough to allow for many operations but also small enough to not take up excessive amounts of computational time and resources, considering the size of the problem instances in the problem domains.

These values offer a compromise in terms of n_k and the number of iterations. The n_k value is near the midpoint of the range of ant colony sizes and 750 is between the generally better performing but computationally costly 1000 iterations and the weaker but computationally more efficient 500 iterations. The algorithm will be applied to the CVRP and MSSP domains.

8.8 Summary

This chapter presented a detailed breakdown of the functioning of the HACO-GP, the ant-based generation perturbative hyper-heuristic algorithm. It detailed how the HACO-GP differs from the HACO-GC in its operation and explained the overall methodology of the generation perturbative hyper-heuristic. The next chapter will present how three HACO algorithms, each with a distinct pheromone map, can be hybridised.

CHAPTER 9

Hybridising Ant-Based Hyper-Heuristics

9.1 Introduction

Developing a hybridisation method that allows for utilising multiple pheromone maps at once, through the use of the HACO algorithm, is one of the goals of this research. This chapter presents the HACOH algorithm, a meta-optimisation algorithm that makes use of three separate HACO algorithms, each with one of the three pheromone map types. This hybrid algorithm is used for the four types of hyper-heuristics considered in this research. The rest of this chapter is organised as follows. Section 9.2 presents the hybridisation method used to combine the effects of the different pheromone maps in separate HACO algorithms. Section 9.3 discusses the application of meta-optimisation to the hybrid algorithm. Section 9.4 presents the method by which the meta-optimisation can be achieved. Section 9.5 presents the control parameters for the hybrid algorithm. Section 9.6 discusses how the input requirements for the hybrid can be generated. Section 9.7 presents the experiments that will be done with the HACOH algorithm and finally a summary of the chapter is given in Section 9.8.

9.2 Hybridisation Method

The general idea for the HACO algorithm hybridisation is to make use of separate ant colonies, each with a different pheromone map. So there is a separate ant algorithm with a 1D, 2D and 3D pheromone map respectively for three colonies in total. The HACOH algorithm operates at a meta-heuristic layer and provides how each of the separate algorithms can be executed. Subsequently, it also enables how the pheromone information produced by each of the HACO algorithms can be shared with the other algorithms. Rather than hybridising all of the pheromone maps into a single map, the HACOH algorithm creates a hybridisation of separate HACO algorithms that combines the search potential of all the pheromone maps but with each remaining separated from the others.

A list is used to decide, during a run, when to execute an iteration with which type of ant algorithm (and therefore using which type of pheromone map). The size of the list is equal to the number of iterations allowed for the problem in totality. Each element of the list is a number from 1 to 3 indicating to perform an iteration with the respective ant algorithm at that iteration.

An example of a small list, for five iterations, is given below:

<2,2,2,3,3,1>

In the example, the first three iterations of the run are performed with the 2D pheromone map (2D HACO), the next two are done with the 3D iteration map (3D HACO) and the final iteration is done with the 1D pheromone map (1D HACO). This could be compared to a non-hybrid HACO algorithm that executed six iterations with just a single type of pheromone map.

The important aspect of the hybridisation process is that after every iteration, regardless of the pheromone type, the best path found by the ant which finished its iteration is used to deposit pheromone values in the corresponding pheromone map of the other ant. In this way, the ant algorithms share information about the results of their searches in their respective spaces with each type contributing to an overall search of the entire space.

This process of sharing information across algorithms is similar to the process of the ants within an algorithm sharing information about their respective searches. In theory, this meta-optimisation algorithm could make use of any combination of HACO algorithms with any combination of pheromone maps (such as using all 1D pheromone maps) but testing if using the three distinct types in concert would confer any benefit over a non-hybrid approach is the objective of this part of the research. The hybrid algorithm, therefore, makes use of separate but parallel searches of the same heuristic space. These different pheromone maps represent the same heuristic space but are delineated by their dimensionality (1D, 2D or 3D).

This hybrid process also means that every iteration in the hybrid algorithm is analogous to an iteration in one of the separate algorithms with the difference being that the information of that iteration is shared amongst the algorithms that did not execute for that iteration. The size of the list is given as the number of iterations allowed for a given run of the hybrid algorithm. This would be equivalent to the number of iterations for the non-hybrid algorithms. The hybrid model is therefore called hyper-heuristic ant colony optimisation hybrid or HACOH.

9.3 Meta-Optimisation

Some kind of optimisation is needed to determine the optimal configuration of 1D, 2D and 3D iterations to produce the best hybrid model for a given problem. To that end, an iterated local search (ILS) algorithm is employed on top of the heuristic optimisation. In this way, there is a meta-optimisation process whose output is a solver (represented by the list) representing a combination of three HACO algorithms that can be used for the four kinds of hyper-heuristics. This process is depicted in Figure 9.1.

In the model, the fitness information from the problem is passed upwards to every precursor layer, enabling optimisations that drive further improvements in the fitness. The development of the appropriate list for hybridising the three HACO algorithms with their pheromone maps is therefore a meta-optimisation layer, to the underlying heuristic layer. Each HACO algorithm (and

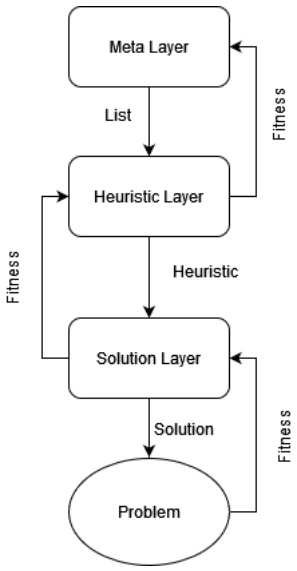

Fig. 9.1: Model of Hybridisation

its pheromone map) interfaces with the underlying heuristic layer but shares its information with the other algorithms. This strategy is generalised across any kind of hyper-heuristic so this general model can be employed by all four of the hyper-heuristics considered in the course of this research.

9.4 Optimisation Strategy

The iterated local search optimisation strategy is given by Algorithm 16. In this algorithm, a list is initialised through random generation, Line 1. It is then evaluated by the hybrid ant system. This evaluation consists of using the specified list in a run of the hybrid ant system. The fitness at the end of the run is taken as the fitness of the list.

The *moveAccept* function is used to update the prior fitness value, $priorFit$ and the prior list, $tmpP$, before starting the iteration process at Line 6. The algorithm will perform some specified iterations, returning the best list as the solver. A memory of the generated lists is used to prevent reusing previously generated lists.

Algorithm 16: Iterated Local Search

Input: h the hybrid ant system, max the max iterations, mp the rate of perturbation

Result: $bestList$ the best list for the hybrid ant system

```
1  init(tmpS);
2  memory = ∅;
3  bestFit = inf;
4  bestList = ∅;
5  fit = h.eval(tmpS);
6  priorFit = fit;
7  moveAccept(tmpS, tmpP, fit, priorFit, bestFit, bestList, mp);
8  it = 0;
9  while it < max do
10 |  tmpP =perturb(tmpS, memory, bestList);
11 |  fit = h.eval(tmpS);
12 |  moveAccept(tmpS, tmpP, fit, priorFit, bestFit, bestList, mp);
13 |  it+ = 1;
```

In this way, the search algorithm will gradually refine an optimal list configuration although, because of the nature of the evaluation, this process is far more computationally intensive than the non-hybrid version of the algorithm. The biggest cost to this is the entire execution of the specific ant algorithm.

9.4.1 Perturbative Function

The perturbative function, given in Algorithm 17, is used to produce a new list from an existing one.

Algorithm 17: Perturbation Function

Input: $tmpS$ the current list, $memory$ the list of known lists, $bestList$ the best list

Result: $tmpP$ a new unique list

```
1  tmp = tmpS;
2  while memory.contains(tmp) do
3  |  tmp =pOperator(tmpS, bestList);
4  tmpP = tmp
```

This function will return a unique list not seen in the memory thus far. If the size of the list is under 10, all of the unique lists could be determined relatively quickly as the number of possibilities is relatively small (2^{10}). In this case, an additional condition can be applied to end the loop if all possible lists have already been generated where the new list is chosen randomly from the memory.

9.4.2 Perturbation Operator

The perturbation operator, the mechanism by which a list is modified, Line 3 of Algorithm 17, is given by Algorithm 18. This operator is the mechanism that produces the new list from the existing one.

The perturbation operator starts by calculating the *numIndices* variable. This variable is calculated using the *mp* variable and indicates the number of indices in the list that needs to be modified. On Line 7, *modList* is initialised to hold all the indices of the current list and this is randomly shuffled.

The *mtList* vector is then populated by some indices based on the *numIndices* variable. This has created two sets of indices. The *mtList* vector contains all of the indices in the list that need to be mutated and thus shifted around, Lines 13–19. The *modList* vector on the other hand determines which elements in the list need to be taken from the *bestList*, (Lines 20–21). By combining these two operations, a new unique list will be generated and returned by the operator.

Initially, the list generation process favours new permutations of the existing list. Over time, the *mtList* vector will shrink in size because the *mp* variable will decrease, thus increasing the proportion of the new list that is taken from the *bestList* and applying an exploitative effect to the list as the iterations progress.

After this process, any of the remaining indices in *modtList* are used to transfer over the values at the indices in the *bestList* to *newList*. The effect of this is to transfer over elements from the best list into the current one. The process will terminate when a unique list is generated.

The effect of this operator is heavily tied to the *mp* variable. The *mp* variable controls the degree to which the generated list will be mutated by the operator. When *mp* is high, there will be a lot of mutation, that is modification, in the *newList* variable as most of the indices will shift to something else, facilitating exploration. This mutation enables exploration as the list changes to incorporate new elements.

However as time goes on, and *mp* decreases, increasingly the *newList* will resemble the best-known list, facilitating exploitation as this best-known list increasingly comes to be represented in the generated lists. This process is not total however as some degree of exploration is needed to prevent a total stagnation of the list.

An example of a list is given below

```
1,2,3,3,2,1
```

In this example, the list is made up of six elements. If the first element was in the *mtList*, then it would be converted to the value 2. Otherwise, its value would be taken from the *bestList*.

Algorithm 18: Perturbation Operator

Input: *tmp* the current list, *mp* the rate of perturbation, *bestList* the best list

Result: *newList* a new unique list

```
1  numIndices=mp*size(tmp);
2  newList=∅;
3  do
4      modList= ∅;
5      mtList= ∅;
6      newList=∅;
7      for i<size(tmp) do
8          modList.add(i);
9      shuffle(modList);
10     for i<numIndices do
11         mtList.add(modList.removeFirst());
12     newList=tmp;
13     for i<size(mtList) do
14         if newList[mtList[i]]==1 then
15             newList[i]=2;
16         if newList[mtList[i]]==2 then
17             newList[i]=3;
18         if newList[mtList[i]]==3 then
19             newList[i]=1;
20     for i<size(modList) do
21         newList[modList[i]]=bestList[[modList[i]];
22 while memory.find(newList)==false;
```

9.4.2.1 Move Acceptance The *moveAccept* function from Algorithm 16 is used to determine whether to update the list and thus whether or not the new list is accepted in the search. It is a type of move acceptance technique that is similar to a simulated annealing move acceptance strategy [31] but with modifications. It is given by Algorithm 19.

The move acceptance technique makes use of a parameter called *mp* which refers to the rate of perturbation in determining whether to accept the current state. If the fitness is a total improvement over the best fitness, then the solution is accepted. If it is not better than the best fitness but it does improve on the prior fitness, that is the fitness of the last meta iteration, then it is accepted.

In the part of the conditional logical, Lines 10–13, a random value r is generated in the range $(0, 1)$ and then if r is less than mp, the move will be accepted. Finally, line 14, adds one additional check to incorporate novel lists into the memory.

Algorithm 19: Move Acceptance

Input: $tmpP$ the prior list, $tmpS$ the current list, fit the current fitness, $priorFit$ the prior fitness, $bestFit$ the best fitness,$bestList$ the best list, mp the rate of perturbation

Result: The current list and fitness are updated or not

```
if fit < bestFit then
    bestFit = fit;
    bestList = tmpS;
    tmpS = tmpP;
    priorFit = fit;
else if fit < priorFit then
    priorFit = fit;
    tmpS = tmpP;
else
    r ~ U(0, 1);
    if r < mp then
        tmpS = tmpP;
        priorFit = fit;
if memory.contains(tmpP) == false then
    memory.add(tmpP);
```

9.4.3 List Approximation

One of the issues regarding applying a meta-optimisation layer to an existing hyper-heuristic algorithm is the high cost of evaluating a given list. The cost of this evaluation scales with either larger lists. This would take the form of larger lists that take longer to evaluate.

A solution to this is to develop a smaller list at less computational cost and then expand that list as needed for larger iterations of the hyper-heuristic. This is referred to as list approximation. More specifically, a smaller list is generated by the ILS algorithm and then this list is expanded into a new, larger size that can be used by the HACOH algorithm without incurring the larger costs of development.

The approximation technique is one of circular addition. Once the initial list is generated, elements from it are repeatedly added to the end of the initial list, starting from the beginning again if the end is reached, until the list of the right size is created.

For example, consider a list of size $5 < 2, 2, 3, 3, 2 >$ that is being expanded to size 13. That new list would look like $< 2, 2, 3, 3, 2, 2, 2, 3, 3, 2, 2, 2, 3 >$. The initial list is repeatedly added to itself, element by element, until the correct size is reached. This is not a perfect solution, as the larger the list being approximated, the less representative the approximation will be of a good list, but

this method does enable large lists to be created from smaller lists, that are less computationally intensive to generate.

9.5 Hybrid Control Parameters

The HACOH algorithm makes use of the same parameters as the HACO algorithm, Section 5.7, with the exception that an additional parameter mp is needed for the ILS algorithm operating in the meta layer.

This parameter is the rate of perturbation and it is used to decide to what degree the perturbation operator can modify a list from one iteration to the next. For these experiments, the value of mp will have an initial value of 0.9 with a final value of 0.1. The parameter will be modified using a decay function like those of Equations 5.5 and 5.4.

The reason for this choice is to facilitate a strategy in the ILS that initially favours producing diverse solutions before transitioning towards intensifying the search around a single solution. More specifically, with these parameters, the search process will initially favour generating widely unique lists but over linear time, it will transition towards favouring making slight modifications to the best solution found thus far. Hence, the search behaviour should be general enough to provide a good solution for all problems without requiring specific parameter tuning for each one.

9.6 Meta-Optimisation Parameters

To conduct the experiments described in Section 9.7, it will be necessary to generate the input lists that will be required by the hybrid algorithms. This is the meta-optimisation process required for the HACOH. This generation process is necessary as generating the lists for each instance will be a costly and time-consuming endeavour across all instances and domains.

The parameters listed below will be used for the creation of the initial lists. As stated, an initial smaller list has to be generated first before it can be expanded into the approximation of the larger list for other experimentation. Hence the use of the reduced parameters here. These parameters are n_k, the number of ants, the number of runs, the number of iterations per run and the path limit. The values are:

- n_k: 10
- Number of Runs: 30
- Number of Iterations (ILS): 100
- Number of Iterations (AS): 100

The number of iterations (ILS) refers to the maximum number of iterations allowed for the ILS to operate with. Given that operating the hybrid algorithm is significantly more computationally expensive to run than the non-hybrid versions, a lower number of iterations is required to ensure the algorithm still executes within a reasonable time, with the given computational resources. The

number of iterations (AS) refers to the number of iterations allowed for the hybrid algorithm to operate in the heuristic space. The choice of 100 iterations for both the ILS and AS is reflective of the time-consuming and intensive process required to generate a list. The choice of 100 iterations for the AS is a large enough set of iterations that can be expanded to meet the requirements for comparing the HACOH algorithm to the HACO without the computational cost required to generate an initial list with an equal amount of iterations to that used by the HACO algorithm.

For the path limit p_l the limit is either taken as the size of the problem for the selection hyper-heuristics or 10/50 for the generation constructive and generation perturbative hyper-heuristics respectively. The reason for this split is the generation constructive hyper-heuristic is building a control function that is applied to the problem whereas the generation perturbative hyper-heuristic builds heuristics that must directly manipulate the problem to modify them and thus merits a larger limit.

The choice of the number of ants n_k is based on the number of ants chosen for the prior HACO algorithms. The intent here is to establish parity between the operating conditions of the HACO algorithms used by the HACOH algorithm and the HACO algorithms that operate outside of this context.

These lists are generated with a single large instance from each of the datasets for each of the domains. The idea is to generate a single list that can be used for all of the instances in the benchmark set. As the list is generated before the testing process, this saves enormously on computational effort as opposed to generating a new list for each instance. It also means that the experimental procedure will be the same as a non-hybrid algorithm as the hybrid will use the list to determine which type of HACO algorithm (with which pheromone map) as it executes its iterations in the same way as the non-hybrid algorithm. The difference is that the non-hybrid algorithm only uses a singular type of pheromone map during its execution whereas the HACOH can draw upon HACO algorithms with all the types of pheromone maps. This is how the HACOH can be compared to the non-hybrid HACO algorithms.

The instances for each of the domains are as follows:

- CVRP: A-n80-k10
- 1BPP: u250_00
- MSSP: C_S2_I2
- QAP: sko64

In the case of the MSSP domain, the same instance is used across all of the hyper-heuristics. To introduce additional robustness, each evaluation of a list will take the average fitness of several runs (three in this case) to minimise the effect of randomness on the results. That is, each list's fitness will be the average of three runs of that list instead of the normal single run.

Of course, the more computational resources that are used during the meta-optimisation process, the greater the potential for improved results. However, this will come with an increased computational cost. If the costs of the meta-optimisation are too large, it becomes more efficient to simply test for the best

pheromone map type to use in a non-hybrid HACO algorithm and this process is designed with that in mind.

9.7 Experiments and Parameter Tuning

As the lists for the HACOH are generated before the experiments, the HACOH then functions like another HACO algorithm for experimentation. The exception of course is that during each iteration, the HACOH will execute one of three HACO algorithms (each with a different pheromone map) instead of using the same pheromone map as the non-hybrid HACO algorithms.

In terms of experiments, the HACOH algorithms are going to be subject to several simulations for each type of hyper-heuristic. These experiments are going to use the lists generated by Section 9.6. The parameters for these experiments are the same as the ones for their respective non-hybrid hyper-heuristics. Thus the HACOH can use the same experimental parameters as its non-hybrid counterpart for each experimental trial. For example, the HACOH-SC algorithm will use the same parameters as the HACO-SC algorithm and so on. Refer to Sections 6.4, 7.9 and 8.7 for more details.

These values were chosen to provide a large enough sample of data for the analysis. The number of iterations afforded to the HACO-GC algorithm is smaller than the HACO-SC and HACO-SP due to the increased computational effort required by the HACO-GC process. The choice of the number of ants is based on available computing resources. The algorithm will be applied to all of the domains of their respective non-hybrid counterparts.

9.8 Summary

This chapter presented the implementation details related to the hybridisation method used to hybridise the different pheromone maps into a single algorithm, the HACOH algorithm. It discussed various details related to the operation of this algorithm from its basic operational structure, to optimisation strategies and experiments. The next chapter will present the results of the experiments conducted for this research.

CHAPTER 10

Results and Discussion

10.1 Introduction

This chapter presents the results of the experiments conducted for this research. Assessment criteria defined in Section 3.5.1 are used to report the results with methods of analysis used to analyse and interpret their meaning. The rest of the chapter is organised as follows. Section 10.2 presents the comparison of the hyper-heuristics in terms of optimality. This comparison is made against the algorithms using the 1D, 2D and 3D pheromone maps as well as the hybrid algorithm. Statistical tests discussed in this section are presented in Appendix A. Section 10.3 provides the comparison of the ant-based hyper-heuristics with their different pheromone maps (including the hybrid algorithm) in terms of the generality metric, the SDD score. Section 10.4 presents an analysis and comparison of the hyper-heuristics in terms of their runtimes. Section 10.5 provides an analysis of the pheromone maps produced by the different algorithms in the different domains. Section 10.6 presents a comparison of the HACO and HACOH algorithms against existing heuristics in the appropriate domain. Finally the Section is concluded in Section 10.7.

10.2 Optimality Assessment of Different Pheromone Maps

This section presents an assessment of the different ant-based hyper-heuristics using different pheromone maps (1D, 2D and 3D) and the hybrid HACOH algorithm. The purpose of this analysis is to determine whether the type of pheromone map played a significant role in improving the performance of the hyper-heuristic in question.

In terms of the assessment criteria for this section, the process makes use of the methods presented in Section 3.5. The best and worst values, average and standard deviation for each algorithm over 30 runs are presented.

This section will primarily consider the effectiveness of the different pheromone maps in terms of their optimality. Additional analysis will be presented in later sections. For reference, Avg and Std Dev refer to average and standard deviation, which are additional metrics added to help contextualise the performance of the different algorithms.

Given the scale and scope of this section, some terms need to be clarified. Firstly, reference to the ant-based hyper-heuristics will be prefaced by the type of pheromone map that the algorithm uses. 1D, 2D and 3D refer to 1D, 2D and 3D pheromone maps respectively. Therefore 1D HACO refers to the HACO algorithm that makes use of a 1D pheromone map. The hybrid algorithm which makes use of separate HACO algorithms with their pheromone map will be referred to as the HACOH. Secondly, when a hyper-heuristic is used in a specific domain, it can be useful to condense the type of hyper-heuristic. For example, GP-CVRP is a generation perturbative hyper-heuristic in the CVRP domain.

10.2.1 SC-QAP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the SC-QAP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.1.1 Results Comparison Table 10.1 provides a breakdown of the results by instance for the ant-based hyper-heuristics and their pheromone maps in the SC-QAP domain. The 3D HACO has the best result in the most number of instances as indicated in Table 10.1.

In terms of the broad strokes, this seems to indicate that the 3D HACO is superior to the other types but there are some things to consider. Firstly, the 3D HACO is not universally superior in all instances. While its aggregate performance is good, there are several instances where the results show that another type of pheromone map is preferable.

For example, els19 is the instance with the largest objective value by magnitude. This instance is an outlier in terms of the range and magnitude of its fitness values in comparison to the problem size. The instance is one of the smaller ones but has the largest fitness range of any of the instances which can be more difficult to solve. In this case, the 1D HACO on average produces a result that is much better than any of the other algorithms. Similarly, for kra30a, kra32, scr12, chr15c and chr18b. All of these problems are relatively medium-sized in the benchmark and the 1D HACO does better on them.

By contrast, the 2D HACO has a more variable spread with regards to where it excels in the benchmark. Of note is the fact that the three largest problems are where the 2D HACO has the best average value, although the margins between the algorithms are much smaller.

The hybrid, HACOH, only has the best value on three instances, chr20c, had20 and sko49. So it seems that the hybrid method is suboptimal here in comparison to using the 3D HACO. The HACOH is also worse than the 1D and

2D HACO algorithms. Of course, relying purely on averages can be misleading, hence the statistical testing performed in the next section.

Table 10.1: SC-QAP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	Best	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
chr12a	9552	14326	10826	17908	1966	12737	10786	15298	1156	**11861**	9552	14286	1378	12750	10786	14442	1104
chr12b	9742	17689	11464	23304	3269	**13288**	11176	15680	1370	13421	11260	15948	1313	14506	12172	15854	1128
chr12c	11156	16613	14332	20684	1814	**13754**	12308	16074	1159	14478	12446	15938	956	14375	11770	15898	1203
chr15a	9896	21276	12806	34368	6408	21410	18536	23412	1475	**21018**	16270	24338	1776	21093	18196	24584	2117
chr15b	7990	25639	21324	32094	3272	20312	14142	24566	2433	**20064**	15494	23890	2227	21005	16328	24554	1978
chr15c	9504	**18635**	13444	28774	5079	19050	15798	21882	1650	19852	17268	22720	1610	20360	16074	23004	2257
chr18a	11098	31156	22572	43866	5360	31256	24988	34928	2899	**29742**	27422	32758	1577	31585	26020	34786	2578
chr18b	1534	**1619**	1534	2034	163	2099	1952	2364	124	2105	1984	2258	71	2142	1870	2316	125
chr20a	2912	5896	4324	6974	693	**5141**	4344	5442	330	5195	4484	5678	400	5363	4982	5930	272
chr20b	2298	5240	4282	6880	745	**4927**	3664	5584	494	4944	4308	5502	333	5165	4098	5928	407
chr20c	14142	53675	26304	68086	12188	50687	41252	55078	3518	51789	45132	58178	3855	**49824**	42698	55414	4269
chr22a	6156	8433	7646	10128	630	8393	7914	8878	279	**7997**	7554	8428	261	8372	7860	8708	223
chr22b	6194	8767	7546	9824	552	8727	8366	9194	247	**8427**	8008	8862	239	8525	7942	9092	335
chr25a	3796	11060	8812	13274	1153	10289	9522	11272	483	**10047**	9362	10782	441	10496	9970	10990	378
els19	17212548	**18905494**	17667270	21352158	1105618	20027870	19166704	21078192	600243	20297673	19071250	21346422	664365	20354413	19486438	21243018	573449
had12	1652	1705	1682	1734	13	1692	1680	1704	8	**1692**	1676	1708	9	1695	1684	1710	7
had14	2724	2841	2794	2884	24	**2808**	2766	2846	21	2815	2790	2848	18	2810	2786	2830	15
had16	3720	3914	3862	3988	32	3866	3838	3894	17	**3852**	3808	3888	24	3862	3832	3898	18
had18	5358	5619	5570	5672	32	5563	5524	5602	22	**5561**	5524	5588	17	5566	5522	5596	20
had20	6922	7300	7190	7384	54	7244	7216	7264	15	7243	7186	7278	25	**7240**	7190	7278	24
kra30a	88900	**108663**	103280	114170	2753	109320	106390	111060	1296	109721	108060	110960	1001	109325	106730	111410	1417
kra30b	91420	111617	108290	115830	2067	111623	109530	113750	1106	**110908**	107750	113170	1440	111439	107490	113300	1401
kra32	88700	**110745**	104250	121270	4028	115934	113130	118540	1344	115797	112740	118290	1488	115763	112580	117370	1552
scr12	31410	**34952**	32662	39370	2077	35003	33406	36398	771	35003	33620	35776	611	35064	33512	36382	842
sko42	15812	18257	17776	18550	206	**18151**	17974	18258	87	18161	17924	18318	104	18163	17888	18324	127
sko49	23386	26909	26630	27146	178	26659	26464	26804	117	26647	26534	26760	55	**26626**	26306	26836	164
sko56	34458	39769	38632	40278	408	39448	39152	39634	138	**39408**	39008	39748	190	39505	39094	39754	170
sko64	48498	55420	54650	55862	374	**55039**	54816	55242	115	55050	54718	55228	158	55053	54814	55250	126
sko72	66256	75378	74630	75828	384	**74940**	74654	75272	169	75029	74490	75356	247	75083	74758	75414	206
sko81	90998	103494	102694	104462	459	**102934**	102388	103340	276	103030	102590	103514	220	103065	102568	103354	260

10.2.1.2 Friedman Test The first part of the statistical testing procedure outlined in section 3.5.4 is to conduct a Friedman test. The purpose of this test is to determine the statistical significance of the comparison between the different results. As this is a study of algorithm optimality, the tests are conducted between the instances to determine where the exact differences lie.

Table A.3 provides the outcome of the Friedman Test as applied to the QAP benchmark. In terms of the results, a total of 19 of the total 30 instances were statistically significant. In this case, this test has shown that for the majority of the QAP instances, the choice of which pheromone map to use has a meaningful effect on the quality of the results.

In terms of the instances which failed to reach significance, as indicated by the Do no Reject H0 outcome, the majority of these instances are on the smaller side such as chr15a, chr15c, chr18a and so on. Sko42 and kra30a and kra30b are the larger exceptions but it is apparent that in general, for some of the smaller instances, the choice of pheromone map is less meaningful.

Els19 is one of the smaller problems with 19 elements, but it also has the largest fitness value by magnitude. So partially, the significance of the differences also relates to the complexity of the problem beyond simply the number of elements considered. Nevertheless, the majority of the instances show meaningful differences between the different pheromone maps in terms of their performance.

10.2.1.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison Table A.1 provides the post-hoc testing for the instances where significant differences were found via the Friedman Test.

The first comparison compares the 2D HACO to the 1D HACO. The results show that H0 was rejected in the majority of instances except for a few instances. These are chr18b, chr22a, els19 and kra32. The rejection of H0 indicates that the 2D HACO produced lower (and thus better) fitness values for those instances than the 1D HACO. For some of these instances like els19 and kra32, the 1D HACO produced the best value of any of the methods. However, when examined as a whole, the overwhelming trend indicates that the 2D HACO outperformed the 1D HACO in the SC-QAP domain. The effect size, in this case, is also telling as most comparisons are at least 1.0 in magnitude, indicating a relatively large difference between the two pheromone maps.

The next comparison is made between the 2D and 3D HACO algorithms. In this case, all of the tests did not reject H0, and thus, this indicates that the 2D HACO performed worse than the 3D HACO for all of the instances where the differences were meaningful. The effect sizes here are relatively, especially in comparison to the first case, which indicates the magnitudes of the difference are smaller, although still significant.

The third comparison is made between the 1D and 3D HACO algorithms. For this comparison, the majority outcome is not to reject H0. There are some cases where H0 is rejected, however. These are chr18b, els19 and kra32. This is similar to the first case where the 1D HACO produced some of the better results for these instances. The majority of the cases result in H0 not being rejected which strongly indicates that the 3D HACO performed better than the 1D HACO based on the null hypothesis. For most of the instances, the magnitude of the effect sizes is at least 1.0 indicating a significantly large difference between the groups.

10.2.1.4 Post-Hoc Analysis: Hybrid Comparison Table A.2 provides the post-hoc testing comparing the HACOH to the non-hybrid HACO algorithms for instances where significant differences were found via the Friedman Test.

The first comparison made in this test is against the HACOH and 1D HACO. In the majority of instances, H0 was rejected with only four cases where it was not. These are chr18b, chr22a, els19 and kra32. Based on the overwhelming rejection of H0, it is clear that the HACOH algorithm performed better than the 1D HACO. This is further corroborated by the magnitude of the effect sizes between the comparisons, as they are generally large.

The next comparison is made between the HACOH and the 2D HACO. Across all instances, H0 is not rejected and thus, the outcome of this comparison strongly indicates that the HACOH algorithm performed worse than the 2D HACO.

The final comparison is made between the HACOH algorithm and the 3D HACO. In this comparison, again, H0 was not rejected. The outcome here is therefore that the HACOH algorithm performed worse than the 3D HACO based

on not rejecting H0 in all compared instances. The effect sizes for this comparison are smaller than in the first case, showing a closer difference between the HACOH and the 2D HACO and 3D HACO.

10.2.1.5 Analysis and Discussion The results of this testing procedure indicate that there are meaningful differences between the 4 types of pheromone maps considered in this research with regards to their effect on the performance of ant-based hyper-heuristics in the SC-QAP domain.

Based on these results, there is enough statistical evidence to indicate that the best performing pheromone map for the selection constructive hyper-heuristic and domain, is the 3D pheromone map. The 2D HACO performed second best with the worst method being the 1D HACO. Part of the results are skewed by the statistical outlier of els19, but in aggregate, the 3D HACO performs better than the others with the 2D HACO being slightly worse.

These results are unusual because they contradict the hypothesis regarding the utility of the 1D pheromone map and its use in selection constructive hyper-heuristics. The assumption regarding the 1D pheromone map was that its condensation of the heuristic space would be ideal for cases where the hyper-heuristic was operating in an environment with stochastic heuristics making precision heuristic selection more difficult.

What these results indicate is that the 3D HACO is capable of overcoming the problem of non-deterministic construction heuristics and refining a heuristic selection that performs better than the other pheromone maps by the hyper-heuristics for this domain. With a sufficient number of iterations, for a run, the ability to retain more information about the heuristic space could result in a filtering effect that is capable of sifting through what would otherwise be statistical noise due to the non-deterministic construction heuristics.

10.2.2 SC-MSSP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the SC-MSSP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.2.1 Results Comparison In terms of the results presented in Table 10.2, there are several interesting outcomes. Firstly, the 2D HACO and 3D HACO achieve the best result of the algorithms on a number of the smaller problem classes. Specifically, between them, they get the best value for instances in the first and second problem class of the MSSP benchmark. The distribution

of these outcomes would suggest some parity between the algorithms when the problem instances are small.

The second major insight is that the 1D HACO has the best result in all instances starting from the last instance from the second class through all the remaining classes. This carries with it an interesting implication about the nature of the effect that the type of pheromone map has on the results. Namely, when the problem classes are small, either a 2D or 3D pheromone map is generally capable of producing good results when used in the hyper-heuristic. However as soon as the problems grow in scale, then the 1D pheromone map becomes overwhelmingly dominant in the hyper-heuristic for those types of instances.

In general, the 1D HACO produces the best results if viewed over the MSSP benchmark as a whole, but it does better on the larger problems for this selection constructive hyper-heuristic and domain. These results will be inspected more deeply via statistical testing in the next section.

Table 10.2: SC-MSSP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	Best	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
C_S_0_I0	25905	27468	26373	29221	824	26724	25905	27182	352	26792	26247	27538	317	**26635**	25905	27333	386
C_S_0_I1	19829	21867	21074	23136	558	21479	20720	22070	333	**21199**	20191	21728	389	21460	20191	21836	480
C_S_0_I2	22224	23370	22340	24977	751	**22625**	22224	23504	415	22632	22289	23017	236	22780	22224	23374	381
C_S_0_I3	25531	26813	25531	28180	591	**26699**	26246	26995	252	27015	26119	27595	334	26734	26238	27565	346
C_S_0_I4	19240	20595	19773	21638	550	20101	19799	20692	224	**20028**	19588	20409	215	20183	19831	20611	204
C_S_1_I0	88856	99844	93444	105570	3261.5789	**99716**	96154	101558	1442	100093	96517	104001	1779	100691	98942	103676	1182
C_S_1_I1	116974	130007	121366	135726	4007	**128357**	123634	131108	2141	128586	124144	131956	2515	129672	126769	131804	1625
C_S_1_I2	66375	77499	74974	82514	2004	75781	73330	78202	1276	**74799**	72032	77141	1726	75270	71021	78197	1840
C_S_1_I3	56796	66501	61913	69841	2473	63568	61138	65708	1542	**63049.4**	60395	65927	1404	63944	58760	66182	2115
C_S_1_I4	82323	**93868**	88355	99363	3178	94952	91858	97528	1670	94017	91156	96094	1293	94829	89903	97942	2221
C_S_2_I0	613810	**760780**	678420	814588	34344	766461	755417	778735	6358	766660	746132	778081	9617	771648	751923	781077	7400
C_S_2_I1	511921	**656832**	590466	706732	43644	678719	655977	690827	8921	677368	653338	695634	11574	679943	659483	692178	10421
C_S_2_I2	660073	**791406**	683718	884958	64683	832518	817300	841749	6656	829073	808214	843909	9949	832449	816975	844509	6674
C_S_2_I3	668031	**792233**	715597	909702	58244	894537	876299	902355	7262	888342	853259	905444	12962	896210	887801	903989	5623
C_S_2_I4	547405	**688152**	595529	741850	47026	714653	695966	725523	9067	720940	708394	735133	7655	712937	702060	727171	7388
C_S_3_I0	2291111	**3019516**	2712995	3244447	157889.8715	3136898.733	3106890	3163746	15668	3139648	3092194	3167214	18569	3138025	3100799	3165635	16360
C_S_3_I1	2442290	**3165580**	2778004	3454147	242104	3353435	3324246	3379732	17506	3361643.8	3322998	3397361	23729	3350548	3286919	3389557	30088
C_S_3_I2	2325007	**2979895**	2663189	3160975	153791	3103248	3076596	3118292	14047	3089887	3033328	3142434	29853	3098779	3035000	3122538	24712
C_S_3_I3	2537219	**3279478**	2854790	3476158	198376	3389864	3322181	3436963	26833	3394192	3360224	3425603	16416	3393305	3355733	3431294	23584
C_S_3_I4	2340895	**3116319**	2737901	3306174	183675	3186347	3151536	3216700	17104	3188119	3106744	3243935	36379	3187066	3134330	3217456	24656

10.2.2.2 Friedman Test The results of the Friedman Test are shown in Table A.4. These results are very different from those presented in Table A.3 as only 8 instances show meaningful statistical significance based on the results of the Friedman Test. Therefore, only 8 of the 20 instances are meaningful enough to perform post-hoc analysis on.

While this is an unusual result, most of the meaningful results are found in the smaller problem classes (class 0 and 1). This would seem to indicate that for larger problems, the difference between the pheromone maps is largely irrelevant and that only for the smaller problems, should the choice of pheromone map be an important decision.

10.2.2.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison The results of the post-hoc analysis for the SC-MSSP domain are presented in Table A.5. For the first comparison, between the 2D HACO and 1D HACO, the

majority of instances reject H0, indicating that the 2D HACO has better results than the 1D HACO. However, as noted in the prior section, these instances are primarily in the smaller classes. For the larger problems, C_S_2_I3 and C_S_3_I2, H0 is not rejected showing the utility of the 1D pheromone map over the 2D. The effect sizes in this comparison also tend to be at least around 1.0 in magnitude, demonstrating large differences between the two groups.

The second comparison is made between the 2D HACO and 3D HACO. In this case, H0 is not rejected in all instances which very clearly indicates that is worse than the 3D HACO. The magnitudes of the effect sizes are closer to the medium level as well, which shows a weaker but still present distinction between the two pheromone maps used in the hyper-heuristics.

The final comparison is made between the 1D HACO and 3D HACO. In this case, H0 is not rejected for all the instances except for the last two, both of which are from the larger classes. This indicates that the 1D pheromone map has worse results on the smaller problems, but for larger problems, the 3D pheromone map is better. The effect sizes are also similarly large.

10.2.2.4 Post-Hoc Analysis: Hybrid Comparison The results of the post-hoc analysis for the SC-MSSP domain, in terms of comparing the hybrid to the non-hybrid methods, are presented in Table A.6.

The first comparison is made between the HACOH and the 1D HACO. These comparisons result in varied outcomes. Half of the outcomes were to reject H0 and the other half resulted in not rejecting H0. In this case, the majority of the rejections of H0 are centred around the smaller problem classes. These results indicate that the differences between the 1D HACO and the HACOH are highly variable. In some cases, the 1D HACO will do better but in other cases, it will not.

In the second comparison, between the HACOH and 2D HACO, H0 is not rejected in all cases which indicates that the HACOH performed worse than the 2D HACO. The effect sizes are again around the medium level for most of the instances.

The third comparison between the HACOH and the 3D HACO also has a similar outcome in that all of the outcomes were to not reject H0 in all instances. This indicates that the hybrid failed to beat the results of the 3D HACO. The effect sizes for this comparison are also larger than in the previous case increasing the magnitude of the differences between the two methods.

10.2.2.5 Analysis and Discussion The outcomes of this experiment are complicated in terms of their implications. The most obvious issue is that in this domain and for this hyper-heuristic, most of the comparisons failed to reach statistical significance which indicates that in this domain, the choice of pheromone map is largely irrelevant.

However, that being said, there were still differences between the pheromone maps when used in the hyper-heuristics and these trends are at least partially

instructive in terms of how the pheromone maps should be used for hyper-heuristics in this domain. Firstly, the use of the 1D pheromone map did produce better results, albeit by a small margin, on the larger problems and so when trying to solve larger SC-MSSP instances, using the 1D pheromone map should be the first consideration.

For the smaller problems, the 2D or 3D pheromone have approximately equal capacities when used in hyper-heuristics across a wide array of these instances with the 3D pheromone map having a slight advantage. So in practice, any type of pheromone map may be used although the 1D pheromone map is the simplest to deploy out of the four types presented here. The 1D pheromone map is also less computationally expensive in comparison to the other pheromone maps when used in the ant-based selection perturbative hyper-heuristic.

10.2.3 SP-QAP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the SP-QAP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.3.1 Results Comparison Table 10.3 presents the results of the experiments with hyper-heuristics using different pheromone maps in the SP-QAP domain. The most immediate implication of the results is that the 2D and 3D HACO algorithms generally both can produce the most optimal solution but for different instances. The 1D HACO and the HACOH are rarely able to compete in terms of producing the best solutions, with the HACOH only having the best result for the sko81 instance.

In terms of the grouping of these results, the 2D HACO tends to produce the best result on the smaller problems like chr15–ch22. It is certainly capable of producing the best result in larger problems like kra30b but generally, the smaller problems are where it does better.

The 3D HACO, by comparison, does do better on the larger problems. For the sko instances, it has the best solution for three out of the five instances. It also finds the best solution on the els19 instance which is one of the more complicated ones in the benchmark. While the 3D HACO can produce good solutions for smaller instances like chr12b, it seems better suited for larger instances.

Table 10.3: SP-QAP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	Best	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
chr12a	9552	13351.8667	11566	15864	1078.631	**11760.9333**	9948	12606	762.1004	11822.2667	9938	12764	822.7719	12446.1333	11190	13258	588.1115
chr12b	9742	14293.6	12220	18276	1548.8101	12256.9333	11048	13986	787.1612	**12119.2**	10102	12978	755.7826	12383.4667	10692	13512	911.3856
chr12c	11156	13383.6	11856	14658	878.6373	**12969.7333**	12396	13884	427.0543	13185.7333	12048	14376	730.683	13328.8	12584	14232	461.7036
chr15a	9896	18855.0667	15860	24488	2005.8645	16326.8	13672	17688	1013.1904	**16016.1333**	14866	17778	807.1967	16285.7333	14934	18096	863.1351
chr15b	7990	18250.5333	14860	21794	2064.172	**15510.2667**	13326	17028	1078.571	15577.3333	13330	16868	870.0698	16097.8667	13586	18062	1400.4106
chr15c	9504	19475.2	14342	23538	2352.5249	**16426.5333**	14978	19230	984.4005	16514.6667	14768	18050	1096.9716	17168.2667	15426	18688	975.3818
chr18a	11098	29251.4667	24874	32436	2127.3975	23853.4667	21904	26018	1251.8588	**22998.6667**	19232	26590	1957.8059	24286.8	20174	27628	2145.4518
chr18b	1534	2169.2	1944	2362	105.9482	**2017.7333**	1880	2074	66.3018	2036	1936	2136	53.4362	2080.5333	2010	2148	45.0109
chr20a	2912	4677.2	4262	5146	257.0801	**4169.8667**	3704	4572	285.7526	4354.5333	4060	4672	172.9087	4348	3984	4734	225.8976
chr20b	2298	4763.4667	4168	5420	347.4828	**4115.4667**	3390	4444	276.9688	4168.6667	3694	4582	272.6271	4365.2	3910	4820	253.9998
chr20c	14142	44532.9333	36228	49874	3833.7586	**37269.3333**	33444	40414	2121.8478	37558.8	28996	41434	3057.8935	38427.6	34756	41812	2027.6274
chr22a	6156	8593.6	8004	9346	374.0406	**8017.2**	7666	8240	156.402	8074.1333	7800	8312	142.6674	8095.6	7668	8392	184.0407
chr22b	6194	8371.7333	7878	8820	253.5867	**7996.4**	7642	8280	180.9107	8006.9333	7778	8282	157.7772	8136.5333	7710	8348	175.1366
chr25a	3796	10481.6	9116	11286	558.8014	**9405.4667**	7938	10104	506.8646	9618	8450	10212	402.9605	9634.2667	8506	10288	502.3931
els19	17212548	21578297.87	20023460	22843744	776643.6208	20082429.47	18846466	21131534	663977.339	**19695913.87**	18877874	20234732	382723.1288	20504867.33	18190174	21911252	873611.6546
had12	1652	1685.6	1668	1706	10.6422	1672.8	1662	1682	6.4498	**1670.8**	1656	1680	6.6676	1673.3333	1662	1688	7.1979
had14	2724	2804.1333	2770	2864	27.2813	**2769.7333**	2756	2780	8.94	2770.9333	2748	2794	15.4155	2778.6667	2758	2792	9.4617
had16	3720	3838.1333	3802	3866	19.9924	**3804**	3768	3830	16.2129	**3804**	3766	3826	17.8406	3818.1333	3780	3840	18.2751
had18	5358	5537.7333	5482	5594	34.7799	5502.8	5458	5524	18.5287	**5501.2**	5456	5538	23.3214	5504.6667	5460	5542	22.9492
had20	6922	7208	7124	7254	31.6047	7145.3333	7082	7182	32.2638	**7138**	7086	7180	27.1504	7159.4667	7094	7202	30.4956
kra30a	88900	113349.3333	111210	116110	1224.0534	110398	107730	112060	1117.4346	**110149.3333**	107450	111610	1578.1475	110405.3333	107440	111780	1169.6389
kra30b	91420	114261.3333	109970	117460	1785.7366	**111251.3333**	107690	113730	1507.8597	111992.6667	109860	114010	1036.0189	112551.3333	110710	113580	873.0559
kra32	88700	109326.6667	107760	111380	1000.7902	107829.3333	106360	109160	889.6827	**107772.6667**	105260	109360	1109.0824	108130.6667	105060	109660	1436.2176
scr12	31410	34472.6667	33252	35806	721.0039	**33252.4**	32596	34238	494.1765	33586.9333	32662	34232	495.2984	33295.0667	31410	34288	824.9931
sko42	15812	18411.2	18234	18526	94.1543	**18221.0667**	18160	18306	47.7485	18230	18038	18328	81.6368	18278.6667	18108	18370	73.117
sko49	23386	26935.8667	26730	27084	107.4948	**26738**	26532	26888	121.479	26742.1333	26574	26886	92.048	26783.2	26572	26964	116.0167
sko56	34458	39848.8	39260	40172	234.7446	39661.6	39418	39866	106.4631	**39606**	39430	39792	105.6463	39666.2667	39230	39898	195.3149
sko64	48498	55568.9333	55124	55786	190.4564	55313.4667	54984	55478	168.0586	**55238**	54990	55430	124.3957	55287.4667	55096	55508	140.5712
sko72	66256	75718.2667	75422	76090	175.0508	75381.8667	75002	75736	192.6514	**75322.8**	74982	75564	170.0887	75414.4	74994	75764	236.927
sko81	90998	103565.8667	103012	104010	307.356	103099.7333	102612	103396	220.5912	103126.2667	102668	103440	239.036	**103080**	102694	103498	255.4548

10.2.3.2 Friedman Test The results of the Friedman Test in the SP-QAP domain are presented in Table A.7. In terms of the results, the testing has demonstrated statistically significant differences, the rejection of H0, in all instances save chr12c which is one of the smallest problems. The instance stands out as unusual as H0 was rejected in the other two ch12 instances. However, the overwhelming majority of the instances showed significant statistical differences.

In general, the p values for these tests are large, well below 0.05. This indicates a very strong rejection of H0 and gives strong evidence for there being differences between the effectiveness of applying the different pheromone maps. The post-hoc analysis presented in the next section will clarify those differences.

10.2.3.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison The results of the post-hoc analysis for the SP-QAP domain are presented in Table A.8. The first comparison is made between the 2D HACO and 1D HACO. There is a conclusive rejection of H0 across all instances in the comparison. This indicates that the 2D pheromone map performed better than the 1D pheromone map when used in hyper-heuristics across essentially the entire QAP benchmark. Furthermore, the effect sizes also indicate very large magnitudes of differences between the two groups most of the effect sizes are at least larger than 1.0 which is a large effect size for this comparison.

In the second comparison, between the 2D HACO and 3D HACO, all of the outcomes are to not reject H0. This would indicate that the use of the 2D pheromone map is either greater than or equal to the 3D pheromone map when used in hyper-heuristics in the majority of the instances in the dataset. The effect sizes here are very useful because they are generally much smaller than in the first case. This would indicate that while there are some stronger differences, such as chr20a, where the differences between the 2D and 3D pheromone maps are more pronounced, in general, they are far more equivalent than apart.

Finally, the third comparison is made between the 1D HACO and 3D HACO. In this comparison, H0 is not rejected in all instances as it was in the previous case. Unlike the prior case, the effect sizes here are much larger, all being at least larger than 1.0. Therefore this gives strong evidence to the notion that the 1D pheromone map is much worse, in terms of optimality than the 3D pheromone map when used in hyper-heuristics in the majority of instances in the SP-QAP benchmark.

10.2.3.4 Post-Hoc Analysis: Hybrid Comparison Table A.9 presents the results of the post-hoc analysis of the hybrid compared to the non-hybrid algorithms for the SP-QAP domain. In terms of the outcomes, the results closely mirror that of the prior testing. Firstly, between the hybrid and 1D HACO, the rejection of H0 in all instances indicates (in conjunction with the generally large effect sizes) that the HACOH performed better than the 1D HACO.

Secondly, in the comparison between the HACOH and the 2D HACO, H0 was not rejected for any instance. The effect sizes are smaller than in the first case but still generally large and so the HACOH algorithm failed to beat the 2D HACO in terms of optimality in the SP-QAP benchmark.

Finally, the third comparison between the HACOH and 3D HACO shows a similar outcome to the second comparison with no rejection of H0 occurring for any instance. The effect sizes are generally also substantive indicating as well that the HACOH failed to outperform the 3D HACO in the SP-QAP domain.

10.2.3.5 Analysis and Discussion The results of the SP-QAP experiments are interesting for several reasons. Firstly, the 1D pheromone map is the worst performing of the pheromone maps when used in hyper-heuristics for this domain. This is followed by the HACOH delivering the next worst performance in terms of optimality.

One of the assumptions regarding the 1D pheromone map was that it would be better able to manage in an environment where stochastic heuristics make feedback information from heuristic selections unreliable. However, the data has not borne this out and instead the opposite is true. The 2D and 3D HACO algorithms generally perform better in this domain than the 1D HACO or HACOH.

Rather it seems to be the case that the additional information-carrying capacity of the 2D and 3D HACO algorithms can help deal with the potentially stochastic perturbative heuristics. However, the difference between these two pheromone map types when used in hyper-heuristics is much more marginal.

For smaller problems generally using the 2D HACO seems to deliver good results whereas the 3D HACO tends to do better on the larger problems. On pure aggregate, the 3D HACO does deliver better results across the entire benchmark but this is offset by strong 2D HACO performances amongst the smaller instances. This has demonstrated that there are suboptimal choices for the pheromone maps in the SP-QAP domain, but that the choice of the optimal pheromone map is much more difficult to discern.

10.2.4 SP-MSSP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the SP-MSSP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.4.1 Results Comparison Table 10.4 presents the results of the experiments with ant-based hyper-heuristics using different pheromone maps in the SP-MSSP domain. These results show that generally, the 1D HACO and HACOH are on the worse side of optimality as they rarely produce the best result for the majority of instances. The HACOH did get the best results on the two largest instances C_S_3_I3 and C_S_3_I4 as well as the smallest instance C_S_0_I0 but in general, it struggles in this domain.

Therefore the two strong contenders are the 2D HACO and 3D HACO, as was previously the case in the SP-QAP domain. The 3D HACO does particularly well on the third class of problems while the 2D HACO does better on the first two problem classes. In general, the two methods are relatively close in performance against one another.

Table 10.4: SP-MSSP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	BEST	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
C_S_0_I0	25905	27189.8667	26325	28045	509.2921	26348.7333	25967	26753	238.7307	26377.6	25905	26828	261.6961	**26348.6667**	25905	27080	397.5719
C_S_0_I1	19829	21120.1333	19829	21751	553.877	**20369.8**	20080	21060	316.6213	20441.6	19829	20971	321.6075	20444.4	19829	21052	404.7284
C_S_0_I2	22224	22718.2667	22224	23528	418.5711	**22296.2**	22224	22388	47.8856	22308.9333	22224	22388	52.9966	22321.9333	22224	22388	61.6516
C_S_0_I3	25531	26314.8	25664	26564	275.5001	25747.7333	25531	25862	105.3616	**25686.4**	25531	25933	129.6885	25835.6667	25531	26239	199.6013
C_S_0_I4	19240	20272.8	19836	20770	252.1143	**19483.8667**	19240	19833	198.7838	19583.2	19240	19836	219.8074	19557.5333	19240	19898	211.7926
C_S_1_I0	88856	98926.8	96816	101781	1206.4319	96470	94602	97796	1032.9478	96007.4667	94130	98676	1214.0065	**95977.6667**	92287	98379	1783.4201
C_S_1_I1	116974	131052.5333	129335	132869	1148.703	**127328.4**	122997	129763	1882.3003	127423.3333	123652	129332	1568.8867	128162.4	126430	129259	810.948
C_S_1_I2	66375	72224.7333	68634	74715	1928.8483	**69672.1333**	68121	71626	1064.182	69757.8667	67992	70965	967.5343	69988.6667	66741	71644	1394.8741
C_S_1_I3	56796	63258.3333	59872	67611	2234.5733	60960.4	58515	62654	999.5871	**60582.3333**	59540	61147	470.1492	61243.8	58996	63496	1397.5288
C_S_1_I4	82323	92565.6	88307	95599	1695.0915	89496.1333	87716	90839	985.988	**88827.8**	86720	90807	1109.4062	90187.2667	88450	91278	852.0984
C_S_2_I0	613810	707042.8	687561	719453	8761.8527	**696522.4667**	686315	702720	4273.1914	697855.1333	688880	705280	5001.8518	698776.8	686059	705435	5215.428
C_S_2_I1	511921	605827.2667	591522	618470	7178.4845	597074.1333	589672	601974	3411.4673	**596519.2**	588543	599518	2892.451	598139.2667	592867	602429	2825.4484
C_S_2_I2	660073	721929	709968	736437	8048.4206	712042.1333	709202	714288	1335.1474	**711546.6667**	702987	716708	4010.5049	712056.6667	707988	715942	1967.3728
C_S_2_I3	668031	786377	772943	795017	5283.6429	781158.8667	777880	784474	2161.3576	**779075**	767328	785193	5421.4623	782992.8	770096	786849	4477.7584
C_S_2_I4	547405	640455.9333	625325	646744	7099.7889	631578.9333	624338	637732	3947.8064	**630153.8667**	622031	636522	4165.7064	632086.3333	626255	638429	3938.1288
C_S_3_I0	2291111	2663579.4	2645406	2681691	11114.3934	**2639328.733**	2628154	2651694	6090.2983	2639427.867	2624951	2657156	9907.8482	2645072.467	2611426	2656844	10375.4233
C_S_3_I1	2442290	2813953.667	2789793	2830957	10828.2258	**2788534.733**	2777686	2798679	5869.8223	2790204.933	2780452	2800301	5030.1331	2790025.4	2782182	2796155	4006.0438
C_S_3_I2	2325007	**2633621.333**	2538317	2695693	36624.2042	2639906.2	2597360	2660539	15956.5951	2639216.2	2614590	2657273	13445.5966	2643642.2	2628116	2655992	9487.5216
C_S_3_I3	2537219	2907968.733	2891167	2920026	8695.004	2887622.6	2864390	2902078	9932.1433	2890120.533	2883021	2895284	4339.3175	**2887228.733**	2873013	2902870	8048.1733
C_S_3_I4	2340895	2626296.4	2617731	2636589	5214.3936	2612074.6	2597719	2622755	5495.6773	2614917	2606230	2622665	4311.7584	**2611502.933**	2597719	2624122	9166.7474

10.2.4.2 Friedman Test The results of the Friedman Test for the SP-MSSP domain are given in Table A.10. The results show a consistent trend of rejecting H0, except for the C_S_3_I2 instance where H0 is not rejected. However, for all other instances, H0 is solidly rejected with low p values. This indicates that in the majority of the SP-MSSP benchmark instances, there is a significant difference

between the different pheromone maps when used in hyper-heuristics for this domain.

10.2.4.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison Table A.11 presents the results of the post-hoc analysis for the SP-MSSP domain between the non-hybrid HACO algorithms. The post-hoc analysis in this domain is very similar to the one conducted in the previous SP-QAP domain in several ways.

Firstly, H0 is rejected in all instances in the comparison between the 2D and 1D HACO. This, coupled with the very large effect sizes, strongly indicates that the 2D HACO is better than the 1D HACO in this domain. Secondly, H0 is not rejected in any instance in the comparison between the 2D HACO and 3D HACO. Correspondingly, the effect sizes are much smaller and this more strongly indicates that the difference between the 2D and 3D pheromone maps, when used in hyper-heuristics for the SP-MSSP domain, is relatively minimal. Finally, in the case of the comparison between the 1D and 3D HACO, H0 is again not rejected in any of the instances. However, in this case, the effect sizes are much larger as was the case for the first comparison, and this indicates that the 1D pheromone map is worse than the 3D pheromone map when used in hyper-heuristics for the SP-MSSP domain.

10.2.4.4 Post-Hoc Analysis: Hybrid Comparison Table A.12 presents the results of the post-hoc analysis of the hybrid compared to the non-hybrid algorithms for the SP-QAP domain. The results of the post-hoc analysis are very similar to the prior SP-QAP domain in terms of the outcomes and implications.

The rejection of H0 in all cases in the comparison between the HACOH and 1D HACO, coupled with the large effect sizes, indicates that the HACOH performed better than the 1D HACO. In the comparison with the HACOH and the 2D HACO, however, H0 is not rejected for any instance. The effect sizes in this comparison are also generally medium to strong in their magnitude and indicate that the HACOH is generally (but not massively) worse than the 2D HACO. Finally, H0 is again not rejected for any instance in the comparison between the HACOH and the 3D HACO. The effect sizes are somewhat more variable in this comparison, especially in the first two instances, but large enough to indicate that the HACOH is worse than the 3D HACO for the SP-MSSP domain.

10.2.4.5 Analysis and Discussion The HACO and HACOH algorithms have proven remarkably consistent when applied to the SP-QAP and SP-MSSP domains in terms of their results. In both sets of experiments, the outcomes whilst not identical, are extremely similar. That is, in both experiments, the 1D HACO was the worst-performing method with the HACOH being the second-worst and the difference between the 2D and 3D HACO being relatively close, with a slight edge towards the 3D HACO that is offset by the larger computational requirements for using a 3D pheromone map in the hyper-heuristic.

It was hypothesised during the design of the 1D HACO that the compressed heuristic space would be better able to navigate the selection space because it would be easier to search by its reduced size. Furthermore, the non-deterministic behaviour of many of the heuristics (constructive or perturbative) would hamper the abilities of the 2D and especially 3D HACO algorithms to refine a good heuristic selection as one set of selections would not necessarily be consistent from execution to execution.

In practice, however, these results demonstrate that in fact, the 1D pheromone map is unsuited for this type of hyper-heuristic, consistently demonstrating poor performance regardless of the domain. While the 1D pheromone map could be employed by hyper-heuristics in the SC-MSSP domain, in both selective hyper-heuristic domains, it proved suboptimal, along with the HACOH.

2D and 3D pheromone maps have been demonstrated to be optimal for this type of hyper-heuristic. A large part of this can be attributed to the larger capacities of both (but especially the 3D) pheromone maps to retain and structure heuristic information. With sufficient iterations, these pheromone maps will be able to filter out enough information to derive better more optimal structures of heuristic information. While the 1D HACO and HACOH are not optimal for this type of hyper-heuristic, it may also be true that the real limits between the 2D and 3D pheromone maps are less meaningful as well.

10.2.5 GC-1BPP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the GC-1BPP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Due to a large number of instances, some of the results have been further subdivided by the specific subsets in the 1BPP benchmark for the sake of clarity.

10.2.5.1 Results Comparison Table 10.5 presents the results of the experiments with ant-based hyper-heuristics using different pheromone maps in the GC-1BPP domain.

The results in Table 10.5 that the best performing hyper-heuristic is the HACOH. It achieves the best average fitness over all of the instances in the sub-benchmarks (u120, u250, u500, u1000 and HARD) when compared to the 1D, 2D and 3D pheromone map. The 1D HACO performs the worst across all of the sub-benchmarks with the 3D HACO doing better and the 2D HACO being second. The average value represented in Table 10.5 represents the aggregate fitness value for every instance in the given sub-benchmark so this strongly indicates that the HACOH was the most algorithm for the GC-1BPP domain.

Table 10.5: HACO-GC 1BPP Results by Pheromone Map Type

	1D HACO				2D HACO				3D HACO				HACOH			
Instance	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
u120	1.135	1.037	1.200	0.048	1.028	1.012	1.087	0.022	1.059	1.014	1.121	0.033	**1.022**	1.011	1.074	0.018
u250	1.136	1.022	1.194	0.054	1.029	1.013	1.082	0.020	1.049	1.019	1.109	0.028	**1.022**	1.012	1.059	0.013
u500	1.140	1.027	1.203	0.057	1.026	1.014	1.070	0.015	1.041	1.016	1.099	0.022	**1.020**	1.011	1.043	0.009
u1000	1.127	1.020	1.202	0.066	1.025	1.012	1.072	0.016	1.035	1.014	1.074	0.017	**1.017**	1.007	1.033	0.007
HARD	1.098	1.070	1.142	0.021	1.064	1.041	1.091	0.013	1.078	1.052	1.103	0.017	**1.063**	1.036	1.093	0.015

10.2.5.2 Friedman Test The results of the Friedman Test for the GC-1BPP domain are given in Tables A.13–A.16. Despite the number of instances that comprise the benchmark set for the 1BPP problem, the outcome of the Friedman Test in all results was to reject H0. This heavily indicates that the type of pheromone map is statistically significant when considering which pheromone map to use in hyper-heuristics for the GC-1BPP domain. The post-hoc analysis in the next section provides additional clarity on the nature of the differences between the pheromone maps as they pertain to the GC-1BPP domain.

10.2.5.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison Tables A.17–A.18 presents the results of the post-hoc analysis for the GC-1BPP domain between the non-hybrid HACO algorithms.

Despite a large number of instances being compared, when it comes to the comparison between the 2D HACO and 1D HACO, there is only a single instance that results in H0 not being rejected. That is the HARD0 instance, the first instance of the HARD sub-benchmark set. In all other cases, H0 is rejected. This would indicate that the HARD0 instance is more of an outlier to the general statistical trend of the 2D HACO performing better than the 1D HACO across the entirety of the 1BPP benchmark set. This is further corroborated by the large effect sizes showing which highlight the differences further.

In the second comparison, between the 2D and 3D HACO, the results are much more mixed. In this case, there are 60 rejections of H0 and 30 instances where H0 is not rejected. The majority of instances across the benchmark still result in a rejection of H0. In this case, the rejections of H0 are primarily clustered in the u120 and u500 sub-benchmarks whereas in the u500, u1000 and the HARD sub-benchmarks, it is more common to find H0 not being rejected. This implies that for the smaller 1BPP problems, the difference, when used in hyper-heuristics, between the 2D and 3D pheromone map is much more in favour of the 2D pheromone whereas, when used in hyper-heuristics for the larger problems, the 3D pheromone map is more likely to succeed. Although in totality, the 2D pheromone map is better when used in hyper-heuristics when viewed across the entire benchmark.

In the final comparison between the 1D HACO and 3D HACO, H0 is not rejected for all instances in the 1BPP benchmark. This indicates that the 1D pheromone map is worse than the 3D pheromone map when used in hyper-heuristics for all instances in the benchmark. The effect sizes are also generally

large enough to further indicate that the 1D pheromone map performed poorly in comparison to the 3D pheromone map when used in hyper-heuristics for the GC-1BPP domain.

10.2.5.4 Post-Hoc Analysis: Hybrid Comparison Tables A.19–A.20 present the results of the post-hoc analysis of the hybrid compared to the non-hybrid algorithms for the GC-1BPP domain. As in the prior section, the results are split into multiple tables for clarity.

In the first comparison between the HACOH and the 1D HACO, H0 is rejected for all 90 instances. Therefore it is apparent that the HACOH algorithm outperformed the 1D HACO as H0 was rejected in all instances and the magnitudes of the effect sizes are all quite large. This indicates that the hybrid algorithm was preferable to using a hyper-heuristic with the 1D pheromone map.

For the second comparison, between the HACOH and the 2D HACO, the differences are more pronounced and less one-sided. Specifically, there are 22 instances where H0 is rejected and 68 instances where H0 is not rejected. In totality then, H0 is not rejected in the majority of instances. The implication is that the HACOH is either worse than or equal to the 2D HACOH in the majority of instances. Broadly speaking, the cases where H0 is rejected are distributed throughout the entire benchmark so the effects are not localised entirely to a single sub-benchmark. The effect sizes are generally moderate, especially concerning the other comparisons. Therefore is more likely that the 2D HACO and the HACOH are closer in parity rather than that one is better than the other as in purely aggregate terms, the HACOH achieves the best results on 75 instances out of the entire benchmark. Therefore the 2D HACO comes much closer in performance to the HACOH but the HACOH holds a slight advantage.

For the final comparison between the HACOH and the 3D HACO, H0 is not rejected for 5 instances with the other 85 instances being a rejection of H0. One of these instances is u500_04, which stands as an outlier as no other instances fail to reject H0 in that sub-benchmark. However, in the HARD sub-benchmark, the remaining cases of H0 not being rejected are found. This would indicate that in the HARD sub-benchmark, the 3D HACO is better than, or at least more equal to the HACOH in performance. In all other instances, H0 was rejected and the large effect sizes indicate that the HACOH is better than the 3D HACO by a wide margin.

10.2.5.5 Analysis and Discussion The GC-1BPP domain is in many ways, the most complicated domain that is considered in this research. It has the most number of instances and some of the most complicated (especially in terms of size) problems. Therefore, the task of generating new construction heuristics for this domain is not an easy one. In this case, the 1D HACO proves that it is the least suited of the hyper-heuristics in this domain, with the worst results on average across the entire benchmark, followed by the 3D HACO.

Surprisingly, the 2D HACO is the next best performing hyper-heuristic with the HACOH taking the best position amongst the various algorithms. This is

a surprising result because it was hypothesised that the larger, 3D pheromone map would be able to do better in this domain. After all, it is larger and a more expanded pheromone map would be able to retain (and thus refine) heuristic information. While it does outperform the 1D HACO by a wide margin, which indicates that some information capacity is good, the 2D HACO outperforms it. This indicates that there is a limit to how useful that additional heuristic information is before it becomes more akin to noise than useful information. The 2D pheromone map has more heuristic information capacity than the 1D pheromone map, but less than the 3D pheromone map. Therefore with enough iterations, it can also explore the space as well as the 3D but not be as bogged down searching in the larger space that comes with using the third dimension.

This also contextualises the relative success of the hybrid. As the HACOH makes use of all three pheromone maps in separate HACO algorithms, though not in equal proportions, it can benefit from all three map types. The best input list combination made use of the 2D pheromone map the most, followed by a smaller proportion of the 3D pheromone map with the minority pheromone map used being the 1D pheromone map. While the HACOH does produce the best results out of the given hyper-heuristics, it is highly dependent on the input list. Therefore this also means that in cases where it would be impractical to use the HACOH, the 2D HACO could be used instead.

10.2.6 GC-MSSP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the GC-MSSP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.6.1 Results Comparison Table 10.6 provides a breakdown of the results by instance for the hyper-heuristics with different pheromone maps in the GC-MSSP domain. The results demonstrate that the 2D HACO can find the most optimal value in the majority of the instances in the benchmark. There are some instances where the best value is achieved by the 3D HACO such as C_S_0_I1, C_S_1_I4 and C_S_2_I2. These are however in a minority in comparison to the 2D HACO. It is apparent that, from these results, the best pheromone map for hyper-heuristics in this domain, is the 2D pheromone map, in contrast to the 1D and 3D pheromone maps or the hybrid algorithm.

Table 10.6: GC-MSSP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	BEST	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
C_S_0_I0	25905	26680.4667	25967	27334	260.0266	**26256.6667**	25905	26677	234.4634	26369.5333	25905	26677	255.4922	26350.6667	25905	26677	303.2909
C_S_0_I1	19829	22020.3333	21358	22917	348.1222	21805.0667	21358	22056	253.7224	**21763.0667**	21074	22056	350.7228	21804.7333	21358	22056	301.0625
C_S_0_I2	22224	23744.2667	23149	24132	301.1927	**23117.8667**	22224	23492	337.9913	23229.1333	22766	23592	236.3127	23237.9333	22572	23746	360.1161
C_S_0_I3	25531	27649.5333	26778	28283	389.3364	**27045.4**	26333	27161	204.3879	27090.0667	26778	27223	102.9448	27110.9333	26622	27670	227.7271
C_S_0_I4	19240	20840.8	19853	21228	467.3141	**20105.5333**	19853	20266	171.2103	20140.6	19853	20631	242.3934	20198	19853	20467	140.5393
C_S_1_I0	88856	94150.2667	89847	98535	2333.2572	**92286.8667**	91044	93406	786.0308	92313.0667	88856	93406	1174.8298	92411.1333	89738	93406	1054.5081
C_S_1_I1	116974	121810.4667	117254	125617	2494.2018	**119249.6**	117815	120589	854.8347	119553.9333	116974	121297	1161.527	120047.3333	119078	121905	759.1042
C_S_1_I2	66375	69340.7333	67246	70982	1134.9651	**67888.5333**	66375	68700	822.4993	68520.5333	68080	69258	344.6325	68616.6667	67610	69786	720.6681
C_S_1_I3	56796	62299.0667	57879	64947	1887.5394	**58886.6**	56796	61095	1084.1706	59670.3333	57879	61471	1074.7823	59510.5333	57329	61548	1377.8875
C_S_1_I4	82323	86501.6	83836	89423	1963.275	84244.2	82694	85944	1069.2345	**84057.5333**	82323	85989	934.4482	84417.6667	83445	86963	1087.6191
C_S_2_I0	613810	637598.3333	617950	688305	17415.01	**617786.3333**	613810	621619	1651.0543	621622.8	617424	634095	5602.9701	621564.8667	615746	640923	6813.5889
C_S_2_I1	511921	532996	516213	565822	14692.151	**517635.7333**	511921	519819	2041.155	520339.2	515365	527409	3371.4849	520856.0667	514327	526172	2983.7447
C_S_2_I2	660073	674383.2	668605	691199	7284.3009	668643.7333	665961	669596	1465.4432	**668345.8**	660073	671271	2799.0793	668792.3333	663051	669596	1645.2237
C_S_2_I3	668031	706593.7333	702510	721503	6343.0282	**691919**	668031	703244	10251.8942	699395.8667	682231	704395	5737.6299	697253.8667	674069	703534	7262.2397
C_S_2_I4	547405	578172.4667	564511	588640	6290.528	**560945.0667**	547405	570352	6436.0639	565989.4667	553893	574781	5003.2251	566892.2667	561721	574781	4589.5141
C_S_3_I0	2291111	2407406.067	2313557	2502957	49628.9761	**2339189.733**	2291111	2373461	23678.4968	2368548.267	2341365	2415969	23673.8451	2351415.733	2316792	2407256	25619.0845
C_S_3_I1	2442290	2598316.333	2520637	2695320	48812.5384	**2545334.533**	2450292	2572696	37513.5713	2559232.467	2506401	2588690	24745.3924	2548325.667	2442290	2588949	42644.7945
C_S_3_I2	2325007	2454397	2383548	2528276	49563.1226	**2393725.667**	2325007	2470537	38089.3975	2423113.133	2348127	2471258	30783.9334	2408199.267	2363368	2448739	30473.7859
C_S_3_I3	2537219	2661066.867	2578059	2739867	47642.7365	**2610334.933**	2537219	2664494	30460.3419	2627694.8	2591355	2680545	25372.5341	2625107.333	2568275	2681568	35103.6389
C_S_3_I4	2340895	2470545.733	2410438	2527182	36133.5745	**2423950.8**	2387066	2453341	22349.9809	2429020.467	2340895	2466993	36435.7411	2425415.933	2382867	2453920	22478.2951

10.2.6.2 Friedman Test The results of the Friedman Test for the GC-MSSP domain are given in Table A.21. The results of the testing indicate that H0 is rejected in the majority of instances with two exceptions. In the case of the C_S_2_I2 and C_S_3_I3, H0 is not rejected but these are the only exceptions to the general trend. The implication of this is that the choice of pheromone map is meaningful for the ant-based generation constructive hyper-heuristic applied to this domain. Further analysis of these differences is presented in the next section.

10.2.6.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison The post-hoc analysis comparing hyper-heuristics with the 1D, 2D and 3D pheromone maps is presented in Table A.22. The first comparison is between the 2D HACO and 1D HACO. The outcome of this comparison is the rejection of H0 in all instances. The effect size, in conjunction with the rejection of H0 strongly indicates that the 2D pheromone map outperforms the 1D pheromone map when used in hyper-heuristics for this domain.

The second comparison, made between the 2D HACO and 3D HACO, is more different. H0 is not rejected in the majority of instances. However, there are several particular rejections of H0. These are specified in the third problem class, although there is a single rejection of H0 in the second class and fourth class each. The implication of this is that in general, the 2D pheromone map is better than the 3D pheromone map when used in hyper-heuristics for most of the problem classes except for the third class, where the 3D pheromone map is better suited. The effect sizes associated with the third problem class are also generally the largest indicating the further gap between the two pheromone maps for that class specifically.

Finally, in terms of the comparison between the 1D HACO and 3D HACO, H0 is not rejected in all of the instances. With the large effect sizes, there is a strong indication that the 1D pheromone map is inferior to the 3D pheromone map by a significant degree when used in hyper-heuristics for this domain.

10.2.6.4 Post-Hoc Analysis: Hybrid Comparison The post-hoc analysis comparing hyper-heuristics with the 1D, 2D and 3D pheromone maps is presented in Table A.23. In the first comparison between the HACOH and the 1D HACO, H0 is rejected in all instances except for C_S_1_I1 and C_S_1_I2. Except for these two cases, the majority of instances result in the strong rejection of H0 and coupled with the large effect sizes, heavily indicates that the HACOH algorithm performed better than the 1D HACO.

With the second comparison, between the HACOH and the 2D HACO, the results are less promising for the hybrid. H0 is not rejected for all instances and this indicates that the HACOH algorithm performed worse than the 2D HACO. The effect sizes are generally medium to strong indicating that while the difference is meaningful, it is not overly predominant.

Finally with the last comparison between the HACOH and the 3D HACO, again H0 is not rejected in all instances and this coupled with similar effect sizes indicates that the HACOH is not better than the 3D HACO although the difference is again not predominantly massive although meaningful in most cases. While the HACOH is worse than the 2D HACO and 3D HACO, it is much closer to the 3D HACO in performance than the 2D HACO.

10.2.6.5 Analysis and Discussion The analysis of this experiment is relatively straightforward but unusual in some respects. Firstly, the best performing pheromone map when used in a hyper-heuristic is the 2D pheromone map for this domain. The next best, although still worse than the 2D pheromone map, is the hybrid algorithm, HACOH. The 1D and 3D pheromone maps fared relatively poorly in comparison when used in hyper-heuristics in the domain.

When it comes to generation constructive hyper-heuristics, the heuristic space is relatively large and complicated because the hyper-heuristic has to build a heuristic from low-level components. This process necessarily places great emphasis on how precisely heuristics can be constructed and evaluated and the larger heuristics theoretically require more heuristic space to represent. For that reason, it makes sense that the 1D pheromone map fared poorly when used in this domain. The capacity to represent specific information about the heuristics does not exist in a 1D pheromone map and thus the 1D HACO fares relatively poorly.

However, the relative success of the 2D HACO indicates that the information storage capacity of the 2D pheromone map is more than sufficient for this domain. The native capacity of the 2D pheromone map is more than sufficient for the problem without incurring the additional overheads that would be associated with the larger 3D pheromone map. The 3D pheromone map does perform somewhat comparably when used in a hyper-heuristic, but evidently, the additional capacity is not required for the most optimal outcomes. However, the fact that the HACOH performed the second-best indicates that the combination of the pheromone maps can be useful. While the HACOH does depend on configuring the list for the algorithm, these results show that the hybrid algorithm could

be used in cases where choosing between the various pheromone maps would be infeasible.

10.2.7 GP-CVRP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the GP-CVRP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.7.1 Results Comparison Table 10.7 provides a breakdown of the results by instance for the hyper-heuristics with different pheromone maps used in the GP-CVRP domain. With the exceptions of A-n33-k6, A-n37-k6, A-n46-k7, A-n60k9, A-n63-k9, A-n64-k9, the best value for all instances is found by the 1D HACO. The 2D HACO produces the best value in five instances with the 3D HACO only doing so in a single instance, A-n60-k9.

For the larger problems, in the M sub-dataset, the 1D HACO has the best results on aggregate, with the only exceptions to this being found in the smaller instances in the A sub-dataset. However, these instances are broadly distributed and do not conform to specific patterns or clusters and so can be assumed to be outliers rather than a general performance trend for the other pheromone maps. From this, it is apparent that the 1D HACO performs extremely well in the GP-CVRP domain, far better than the other algorithms.

Table 10.7: GP-CVRP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	Best	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
A-n32-k5	784	**920**	859	1097	62	964	906	1020	27	967	919	1001	26	965	897	1011	35
A-n33-k5	661	**724**	675	772	31	754	744	762	5	753	729	768	10	749	690	765	18
A-n33-k6	742	**827**	789	896	33	**827**	800	842	11	828	802	853	15	829	802	853	16
A-n34-k5	778	**814**	789	861	21	838	820	856	11	842	822	859	11	831	797	853	17
A-n36-k5	799	**862**	820	941	41	886	839	914	18	885	855	916	17	888	841	919	22
A-n37-k5	669	**741**	693	871	62	806	750	843	25	802	752	836	22	773	703	849	36
A-n37-k6	949	1063	1013	1152	41	**1055**	1017	1079	16	1061	1036	1094	16	1065	1027	1115	23
A-n38-k5	730	**823**	806	874	18	854	818	869	14	857	835	871	10	843	807	865	19
A-n39-k5	822	**918**	875	980	33	953	931	981	12	953	933	965	9	951	931	968	11
A-n39-k6	831	**967**	907	1097	58	1026	989	1052	19	1035	991	1063	19	1034	971	1066	23
A-n44-k6	937	**1033**	972	1122	51	1106	1051	1154	27	1106	1060	1136	25	1102	1055	1152	28
A-n45-k6	944	**1083**	1015	1278	76	1150	1095	1202	30	1163	1103	1196	25	1122	1057	1199	49
A-n45-k7	1146	**1351**	1254	1520	74	1398	1352	1439	25	1406	1346	1449	30	1407	1346	1429	23
A-n46-k7	914	1055	987	1143	49	**1041**	1010	1059	14	1045	1015	1062	15	1042	1024	1058	10
A-n48-k7	1073	**1235**	1122	1372	77	1283	1201	1321	30	1288	1240	1336	26	1290	1249	1330	21
A-n53-k7	1010	**1111**	1037	1304	69	1157	1126	1190	21	1156	1092	1199	36	1143	1053	1198	40
A-n54-k7	1167	**1380**	1264	1507	72	1459	1389	1497	29	1464	1431	1495	20	1464	1396	1505	31
A-n55-k9	1073	**1225**	1175	1350	46	1269	1247	1290	11	1270	1234	1302	21	1260	1204	1293	22
A-n60-k9	1354	1594	1530	1651	34	1571	1540	1591	14	**1566**	1495	1610	30	1570	1535	1596	17
A-n61-k9	1034	**1163**	1076	1298	63	1192	1169	1212	13	1199	1128	1229	25	1186	1158	1215	17
A-n62-k8	1288	**1429**	1351	1490	42	1540	1503	1588	27	1554	1510	1605	27	1535	1483	1582	33
A-n63-k9	1616	1919	1766	2139	105	**1898**	1808	1947	36	1908	1849	1949	28	1916	1857	1961	35
A-n63-k10	1314	**1555**	1411	1661	69	1571	1534	1610	26	1570	1492	1606	33	1572	1433	1614	51
A-n64-k9	1401	1696	1562	1785	67	**1694**	1656	1718	20	1696	1671	1721	15	1699	1633	1724	23
A-n65-k9	1174	**1413**	1303	1516	69	1484	1459	1510	17	1487	1420	1517	28	1489	1437	1530	28
A-n69-k9	1159	**1382**	1245	1596	91	1521	1464	1570	32	1525	1450	1564	28	1506	1401	1580	55
A-n80-k10	1763	**2136**	1908	2398	159	2201	2069	2271	52	2205	2023	2288	68	2226	2078	2289	51
M-n101-k10	820	**977**	931	1029	28	999	979	1011	8	995	982	1006	8	991	970	1004	10
M-n121-k7	1034	**1370**	1273	1447	60	1551	1449	1611	48	1575	1525	1621	24	1494	1291	1629	95
M-n151-k12	1015	**1536**	1383	1686	91	1645	1575	1695	34	1655	1582	1688	26	1625	1548	1662	36
M-n200-k16	1274	**1821**	1517	2113	157	2000	1871	2055	54	2001	1917	2062	41	1974	1861	2044	51
M-n200-k17	1275	**1828**	1613	2040	145	1999	1909	2060	41	2029	1971	2056	24	1965	1852	2074	61

10.2.7.2 Friedman Test The results of the Friedman Test for the GP-CVRP domain are given in Table A.24. The results indicate that 22 of the 32 statistical comparisons are significant with 10 instances resulting in not rejecting H0, thus indicating that the differences between the pheromone maps are not significant for those instances.

The cases where H0 is not rejected are distributed throughout the benchmark with little in the way of consistent distribution. It appears that some of the instances are not sufficient to produce a meaningful difference between the pheromone maps used by the hyper-heuristics but since the majority of instances are meaningful, and H0 is not rejected on the largest problems, the differences between the pheromone maps are meaningful overall when used in hyper-heuristics for the GP-CVRP domain.

10.2.7.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison The post-hoc analysis comparing the hyper-heuristics using the 1D, 2D and 3D pheromone maps is presented in Table A.25. For the comparison between the 2D HACO and 1D HACO, H0 is not rejected in all instances. As the effect sizes are

quite large, this is a solid indication that the 2D HACO performed worse than the 1D HACO in this domain.

In the second comparison, H0 was again not rejected in any instance. Here, the effect sizes are smaller than in the first case. It more likely indicates that the 2D HACO and 3D HACO are comparable in their performance.

Finally, in the comparison between the 1D HACO and 3D HACO, H0 is rejected in all cases. This, coupled with the large effect sizes, solidly indicates that the 1D pheromone map is better than the 3D pheromone map when used by hyper-heuristics in the GP-CVRP domain.

10.2.7.4 Post-Hoc Analysis: Hybrid Comparison The post-hoc analysis comparing the HACOH to the 1D, 2D and 3D HACO is presented in Table A.26. H0 is not rejected in any instance for the comparison between the HACOH and the 1D HACO in the GP-CVRP domain. Factoring in the large effect sizes strongly indicates that the HACOH algorithm performed worse than the 1D HACO in this domain.

In the second case, H0 was rejected in the overwhelming majority of cases except for the A-n37-k5 instance where H0 was rejected. The effect sizes are also relatively smaller except for the case of A-n37-k5 where it is quite large. From this, it is apparent that the HACOH algorithm is slightly better than the 2D HACO in most instances except for the singular instance where the 2D HACO is better. Given the single outlier, the general trend is for the HACOH to be slightly better than the 2D HACO.

Finally, between the HACOH and the 3D HACO, these results are more of a mixed bag. H0 is rejected seven times although this is dispersed throughout the 22 instances. The general trend is for H0 to not be rejected which indicates that the HACOH can do better than the 3D HACO, although for a sizeable number of instances, the trend is reversed and the 3D HACO outperforms the HACOH.

10.2.7.5 Analysis and Discussion The results for this experiment are contrary to the initial expectations of performance for the various pheromone maps and their usage in the given hyper-heuristics. Primarily, the hypothesis regarding the use of the different pheromone maps in a generation perturbative hyper-heuristic was that more information-heavy (and therefore descriptive) pheromone maps like the 2D and especially 3D pheromone map would be better suited for a generation task given the nature of constructing a heuristic from low-level components.

The results, however, paint a different picture. The 1D pheromone map is the most optimal for this type of hyper-heuristic based on the performance of the 1D HACO in this domain. The 2D and 3D pheromone maps fair generally much worse when used in the hyper-heuristics, although their performance is quite close when the two are compared. The HACOH algorithm is capable of outperforming the 2D HACO and 3D HACO but not the 1D HACO which consistently produces the best results in the overwhelming majority of instances.

A generation perturbative hyper-heuristic, in this case, differs from the generation constructive hyper-heuristic in that its low-level components are essentially contained operations that are being executed on a single solution that changes states. In theory, different chains of perturbation could end up producing the same final solution since the perturbation process will start with the same solution each time. The reduced heuristic space encapsulated in the 1D pheromone map, therefore, makes it easier to search that space for the perturbative operators that best modify the solution.

In the case of the CVRP, optimisations can occur both inside a route and between routes, and the 1D pheromone map, when used in the hyper-heuristic, determines a distribution of perturbative operators that when applied, produce an ideal version of the original solution.

10.2.8 GP-MSSP

This section discusses the optimality results of 1D, 2D and 3D HACO algorithms and the HACOH as they have been applied to the GP-MSSP domain. This analysis will consist of two parts. The first is an assessment of the results breakdown per each instance, followed by the statistical testing procedure outlined in section 3.5.4. For each instance, the average result (over the number of runs) is given, as well as the best and worst value recorded in the run (min and max) and the standard deviation (Std Dev). The best average values are indicated in bold. Where applicable, the best fitness value is included as well.

10.2.8.1 Results Comparison Table 10.8, provides a breakdown of the results by instance for the hyper-heuristics with different pheromone maps used in the GP-MSSP domain. The results indicate that the HACOH has generally the best results of any of the algorithms in this domain. In particular, the HACOH has the best result for ten instances out of the twenty total.

This is a relatively wide margin in terms of the entire benchmark but there are several considerations. Firstly, the 3D HACO has the best results for most of the first (and therefore smallest) problem class. It also finds the best result in four additional instances (in the other problem classes) after its first three. The 2D HACO also has the best results for three of the instances. The 1D HACO fails to have the best results for even a single instance.

In its totality, the HACOH seems to have the best results out of all of the algorithms in most of the problem classes except for the first one where the 3D HACO does better than it.

Table 10.8: GP-MSSP Results by Pheromone Map Type

		1D HACO				2D HACO				3D HACO				HACOH			
Instance	BEST	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev	Avg	Min	Max	Std Dev
C_S_0_I0	25905	28504.5333	27481	28911	449.5687	**27552.4667**	26534	28370	540.9407	27685	26677	28266	490.2727	27781.2	27182	28355	395.3259
C_S_0_I1	19829	22440.9333	21156	23077	483.7888	21589.4	20791	22144	355.6932	**21267.5333**	20080	22156	536.1969	21346.2	20453	22501	595.9744
C_S_0_I2	22224	23599	22388	24600	588.6667	22947.6	22388	23412	404.4912	**22720.5333**	22340	23377	399.5665	22948.6	22340	23412	442.1693
C_S_0_I3	25531	26224.8	25664	27986	575.4873	25822.4	25664	25862	81.9798	**25756.4**	25664	25862	102.2468	25835.6	25664	25862	69.6694
C_S_0_I4	19240	21769.4	19588	23142	998.7664	20616.8	20039	21408	397.5982	20897.8667	20248	21587	415.1401	**20603.4667**	19670	21581	562.3325
C_S_1_I0	88856	100005.5333	98389	103363	1480.5612	97378.8667	94491	99220	1266.0239	97173.3333	94500	99422	1439.9276	**96916.6667**	93797	99891	1617.3109
C_S_1_I1	116974	132940.4667	127645	136212	2606.3755	132303.8667	130732	133819	886.7535	**131368.0667**	129592	133072	1164.988	132673	130235	134946	1247.0487
C_S_1_I2	66375	72689.7333	70058	74159	926.7692	71203.1333	68155	72490	1146.7445	71134.5333	66559	72825	1724.972	**70990.5333**	69335	72209	748.7499
C_S_1_I3	56796	70285.6	61622	73818	2989.5752	**67008.2667**	63102	69738	1928.0504	68397.2667	65610	70883	1442.0376	67463.5333	64971	69502	1331.4476
C_S_1_I4	82323	93276.7333	89335	94651	1318.8786	92526.0667	89775	94131	1601.8068	92928.6667	89904	94269	1316.1782	**92442.1333**	89775	93942	1664.2
C_S_2_I0	613810	724502.1333	716152	730226	4026.2414	718973.5333	699590	730226	8408.8298	**718957.6**	700392	729661	7862.0177	724892.2667	717524	730345	3142.5742
C_S_2_I1	511921	611387.2	596809	632495	9433.8417	606915.5333	596795	610674	3646.2979	610295.6667	604239	618894	3336.3194	**604470.6**	593175	608817	5123.9824
C_S_2_I2	660073	716712.5333	707760	721737	6176.3604	708958.1333	707359	717138	2955.6109	**708055.7333**	699155	715207	3310.3294	709411.1333	707463	715207	2807.9008
C_S_2_I3	668031	777567.6667	755284	803996	17701.7833	764866.9333	758868	766640	2791.4409	763517.6	749162	766640	4888.4447	**761532.4**	751942	766640	4767.9299
C_S_2_I4	547405	636129.3333	620379	644985	8742.3995	**629694**	616943	636005	5161.9089	633034.9333	623091	641333	5517.2823	630858.5333	615715	641158	8068.5857
C_S_3_I0	2291111	2694416.733	2672859	2707540	11965.0767	2685043.667	2670116	2687854	5152.4215	2689390.333	2678608	2697870	6486.4311	**2678393.067**	2638868	2687854	12844.2467
C_S_3_I1	2442290	2811577.4	2754935	2843688	28628.7196	2777791.2	2770922	2788076	4650.19	2776524.267	2754935	2788127	7458.1314	**2766965.2**	2725297	2776186	14059.1487
C_S_3_I2	2325007	2698004.133	2637982	2730621	24737.0005	2677984.733	2644009	2694862	14112.9282	**2677484.467**	2647013	2706361	16282.3371	2682885.2	2658482	2694801	10750.722
C_S_3_I3	2537219	2916434.933	2859700	2963156	25942.5791	2924928.467	2898220	2929212	8615.4982	2931169.8	2918873	2941896	6453.8977	**2906191.533**	2827939	2929212	25760.1152
C_S_3_I4	2340895	2637987.333	2594996	2658259	19138.9029	2636380.4	2619957	2644369	8379.478	2642963	2639144	2647651	2217.7869	**2629768.667**	2582340	2644369	15094.8186

10.2.8.2 Friedman Test The results of the Friedman Test for the GP-MSSP domain are given in Table A.27. The results indicate that H0 is rejected in all comparisons except for C_S_1_I4. This is the only instance where H0 is not rejected.

This strongly indicates that the type of pheromone map is meaningful across the vast majority of instances in the benchmark set and that the choice of pheromone map is meaningful for the GP-MSSP domain as a whole. These differences are analysed further in the next section.

10.2.8.3 Post-Hoc Analysis: 1D, 2D and 3D HACO Comparison The post-hoc analysis comparing the hyper-heuristics that use 1D, 2D and 3D pheromone maps is presented in Table A.28. There are three comparisons made in Table A.28. The first is made between the 2D HACO and 1D HACO. In this comparison, six instances result in H0 not being rejected but for all other instances, H0 is rejected. In particular, H0 tends to not be rejected in the larger and more complicated instances like C_S_3_I3 and C_S_3_I4. for the smaller instances, in the first and second problem class, it is much less likely to happen.

These results imply that generally, the 2D HACO performs better than the 1D HACO over the totality of the benchmark, but there are specific instances where this is not the case. The effect sizes associated with the instances where H0 is not rejected are typically moderate to strong in magnitude and therefore more readily indicate that the 1D HACO is better than the 2D HACO for those cases. Otherwise, the magnitude of effect sizes is large in the instances where H0 is rejected, strongly suggesting that the 2D HACO outperforms the 1D HACO.

In the second comparison, H0 is only rejected in two instances, C_S_2_I1 and C_S_3_I3. The minimal quantity of rejections, in comparison to the majority of instances where H0 is not rejected, suggests those rejections are more outliers than representative of serious differences. In particular, the effect sizes range from fairly minimal (0.00184) to fairly large (1.0095) with most being of moderate magnitude. This indicates that generally, the 2D and 3D HACO are closely aligned in terms of their performance.

Finally, for the third comparison between the 1D HACO and 3D HACO, H0 is only rejected once in the C_S_3_I3 instance. All other instances result in not rejecting H0. The effect sizes here are generally large, indicating that there is sufficient evidence to accept that the 1D pheromone map is worse than the 3D pheromone map when used in hyper-heuristics for this problem domain.

10.2.8.4 Post-Hoc Analysis: Hybrid Comparison The post-hoc analysis comparing the HACOH against the 1D, 2D and 3D HACO algorithms is presented in Table A.29. For the first comparison between the HACOH and the 1D HACO, there are six instances where H0 is not rejected. For the rest of the instances, H0 is rejected. Largely, H0 is not rejected in the larger problem classes like C_S_2_I1. The implication of this is that for the most part, the HACOH algorithm is better than the 1D HACO but there are some cases in the larger problem classes where the difference is less meaningful. The effect sizes in the cases where H0 is rejected are large and they diminish when H0 is not rejected, further supporting this class. Since the majority of instances favour the HACOH algorithm it is reasonable that it performs better than the 1D HACO.

In the second comparison between the HACOH and the 2D HACO algorithm, H0 is only rejected three times, in the two larger problem classes. In all other cases, H0 is not rejected. For the majority of instances, therefore, the HACOH algorithm is worse than or equal to the 2D HACO. Given that the effect sizes in this comparison tend to be moderate size, it is likely that the performances of these algorithms are roughly equivalent.

This trend continues in the third comparison between the HACOH algorithm and the 3D HACO. H0 is rejected five times in the benchmark with the majority of instances resulting in not rejecting H0. The effect sizes associated with rejecting H0 are particularly large, which shows that for the larger problem classes, the HACOH algorithm generally performs better than the 3D HACO. However, for the smaller problem classes, the differences between the HACOH and the 3D HACO are much less meaningful.

10.2.8.5 Analysis and Discussion The analysis of this experiment is not as straightforward as in the case of the GP-CVRP domain. Firstly, the 1D pheromone map is the worst performing pheromone map, when used in the hyper-heuristics, by a relatively wide margin. This is contrary to the GP-CVRP domain where the 1D pheromone map performed the best. This indicates that the 1D pheromone map is not universally optimal for all the ant-based generation perturbative hyper-heuristics. The MSSP is very structurally different to the CVRP and as it is a problem with fewer feasibility constraints, this makes the heuristic space more difficult to optimally traverse.

So in this case, a higher dimensional pheromone map is better because it enables the hyper-heuristic to refine the operator combination in a more precise way. However, in terms of optimality, the margins between the 2D HACO, 3D HACO and HACOH are much more narrow. In absolute terms, the HACOH does produce the best results on aggregate for the most number of instances,

but in relative terms, the margins between it and the 2D and 3D pheromone maps are not so large as to be truly comprehensive.

A generation perturbative is one of the more complicated types of hyper-heuristic, especially given the form presented here. The nature of the problem seems to influence the performance of the pheromone maps and that should be considered when choosing the right pheromone map for the task at hand.

10.2.9 Optimality Implications and Discussion

While optimality is not the only metric with which to assess the HACO and HACOH algorithms, it is nevertheless an important metric with wide-reaching implications for the use of the ant-based hyper-heuristics that are defined in this research. Optimality in the context of hyper-heuristics generally means the examination of the hyper-heuristics in a very granular way. Hence the comparison of hyper-heuristics on a per-instance basis. Wider statistical trends can still be gleaned from this type of analysis but it also enables the close study of a hyper-heuristic's performance inside of a problem domain.

The first major insight from these experiments is that no universally optimal pheromone map configuration works equally well for every type of hyper-heuristic and every problem domain. The four types of hyper-heuristics, while broadly related in that they work in the heuristic space, are distinct enough that the pheromone maps that work well for one hyper-heuristic in a domain, do not necessarily translate to a different hyper-heuristic or even the same hyper-heuristic in a different domain.

In terms of the selection constructive hyper-heuristics, the 3D pheromone map was the most optimal for the hyper-heuristics in the QAP domain, but the 1D pheromone map is most optimal for the hyper-heuristics in the MSSP domain. The SC-MSSP domain in particular suffered from most of its instances not resulting in statistically significant results when the pheromone maps were compared so that tempers the value of the result in that case.

However, looking across both domains, the 3D pheromone did the best optimality-wise when considering how well the pheromone map did when used by hyper-heuristics in both domains. The 3D pheromone map did the best in hyper-heuristics for the SC-QAP domain and second-best in the SC-MSSP domain. Contrary to expectations, the 3D HACO was able to handle the challenge of the non-deterministic construction heuristics and succeeded at this task, whereas the 1D HACO failed in the QAP domain being the worst method. Selection constructive hyper-heuristics are amongst the simplest form of hyper-heuristics given that they assemble full solutions piece-by-piece, so providing the hyper-heuristic with the 3D pheromone map and this type of problem allows it to minimise the impact of the non-deterministic heuristics by greatly refining the placement of the heuristics in the selection process.

For the selection perturbative hyper-heuristics, the worst-performing method in both the QAP and MSSP domains is the 1D HACO, and by extension the 1D pheromone map. The best performing method shifts between the 2D HACO (better in the QAP domain) and 3D HACO (better in the MSSP domain) so the

overall implication is that while the 1D HACO and HACOH are suboptimal for this hyper-heuristic, the choice between 2D and 3D pheromone maps is much more problem-dependent.

The generation constructive hyper-heuristic experiments also revealed interesting results with regard to optimality. Namely, that the 2D pheromone map was the best for hyper-heuristics in the GC-MSSP domain, but that the HACOH performed the best for the hyper-heuristics in the GC-1BPP domain. This domain has generally revealed something of the success of the HACOH in the ant-based generation constructive hyper-heuristic as it did the second-best in the GC-MSSP domain. Considering its performance in both domains, the hybrid performed, on average, as well as the other best performing pheromone map, the 2D one.

In relative terms, the 1D and 3D pheromone map delivered a poor performance for this hyper-heuristic and this can be attributed to the nature of the extremes. The 1D pheromone map cannot encapsulate information well enough to properly construct a good heuristic because its heuristic space is too compressed. The 3D pheromone, rather than being emboldened by the additional information capacity, struggles to sift through it all and ends up suffering in performance as a result. By contrast, the 2D pheromone map strikes the balance between these two methods and delivers great performance in the GC-MSSP domain, and good performance in the GC-1BPP domain. By also representing a compromise, of a different nature, the HACOH also benefits from this and does similarly well.

Finally, in the case of the generation perturbative hyper-heuristic, a similar outcome is observed except it is between the 1D pheromone map, 2D pheromone and the hybrid. For the GP-CVRP, the best method is the 1D HACO and the second best is the HACOH, showing the primacy of the 1D pheromone map followed by the hybrid algorithm approach. For the GP-MSSP, the best method is the HACOH with the 2D HACO being the second best. Considering both domains, the HACOH does the best in totality, but there is a valid enough case to be made that choosing the most optimal pheromone map for the specific domain in the problem is also viable, given the differences in optimality.

So it is the case then, that from the perspective of optimality, the type of pheromone map used has a meaningful impact on the performance of the ant-based hyper-heuristic. There are cases where the wrong choice results in very suboptimal performance and other cases where the most optimal pheromone map would depend on the exact nature of the problem and its circumstances. Nevertheless, making the choice is an important decision that has to be considered when using these ant-based hyper-heuristics.

10.3 Generality Assessment

The results of the SDD score calculation for the hyper-heuristics by pheromone map type and problem domain are presented in Table 10.9. The scores listed quantify the ability of each algorithm type to generalise in that domain and

hyper-heuristic. The average across all of the domains and hyper-heuristics and the corresponding standard deviation are provided as well. The best values are indicated in bold.

Table 10.9: SDD Scores for each Hyper-Heuristic by Domain

SDD Score	1D	2D	3D	Hybrid
SC-QAP	34.75	31.96	**31.79**	32.51
SC-MSSP	**4.30**	5.32	5.71	5.36
SP-QAP	29.69	**24.97**	25.06	25.85
SP-MSSP	**2.43**	2.61	2.77	2.59
GC-BPP	1.68	**1.32**	1.47	1.49
GC-MSSP	1.11	1.04	1.03	**0.93**
GP-CVRP	**8.19**	10.04	10.24	9.63
GP-MSSP	**2.24**	2.34	2.55	2.29
Avg	10.55	9.95	10.08	10.08
Std Dev	13.62	11.93	11.84	12.24

The SDD scores given in Table 10.9 are an important metric to consider when assessing the effect that different pheromone maps will have on the underlying ant-based hyper-heuristic. In particular, the 1D pheromone map is associated with the best SDD score in the most number of cases out of the assessments (four out of eight). It is also more associated with perturbative hyper-heuristics than constructive ones, although the SC-MSSP is the stand out exception. However, in terms of aggregate SDD score, the 2D pheromone map results in the best result over all of the experiments.

The 3D pheromone map and hybrid algorithm only produce the best SDD score in a single experiment each, SC-QAP and GC-MSSP respectively. Although, in aggregate, they still produce a better SDD score than the 1D pheromone map.

There are several implications from these results. The first is that if generality is a prime consideration for the hyper-heuristic, then the choice of pheromone map will influence this factor. The 1D pheromone map will result in better generality but only in specific domains and only for specific (usually perturbative) hyper-heuristics.

However, if an ant-based hyper-heuristic is to be used without necessarily considering generality as a paramount concern, then the 2D pheromone map is likely to result in the best compromise choice as it will generally produce a good SDD score on a wide variety of domains. The 3D and HACOH SDD scores do not differ by a significant enough margin to indicate that they are that different in terms of generality over the experiments that were conducted.

10.4 Comparison of Algorithm Runtimes

The runtimes of the different hyper-heuristic algorithms in their domains are presented in this section. Runtimes are a complicated matter because so many variables and factors can influence their values. These can range from minor changes to code to the type of hardware that the code executes on. These differences make comparisons with runtimes a complicated matter. However, runtimes can also be indicative of the different algorithm computational costs with a runtime providing a rough measure of the costs of running an algorithm. In particular, as the results are delineated by the pheromone map type used by the hyper-heuristic, these results can be indicative of how the choice of pheromone map influences the runtimes of the HACO and HACOH algorithms. The runtime analysis presented here should be considered a secondary aspect of the analysis alongside the studies of optimality and generality presented in sections 10.2 and 10.3.

To that end, the runtimes presented here are aggregated over the instances in the benchmark sets for the different domains and hyper-heuristics. For example, the runtime presented for a 1D HACO in the QAP domain is the average runtime for a single run of the 1D HACO over all of the instances in the QAP domain. All times are given in minutes. In this way, the runtimes of the different pheromone maps and their usage in the HACO and HACOH algorithms can be compared.

10.4.1 Selection Constructive Hyper-Heuristics

The runtimes for the different pheromone maps used in the HACO-SC and HACOH-SC are given by Table 10.10. The best results are indicated in bold.

Table 10.10: Runtimes for the SC on the QAP and MSSP Domains

SC	1D	2D	3D	Hybrid
QAP	**0.01**	0.03	0.03	0.05
MSSP	**0.01**	0.04	0.04	0.10

In terms of the results, the use of the 1D pheromone map results in the fastest computational times for a single run in both domains. The use of 2D and 3D pheromone maps do not differ significantly for this type of hyper-heuristic, in these domains, with the use of the hybrid algorithm producing the worst times in both domains.

10.4.2 Selection Perturbative Hyper-Heuristics

The runtimes for the different pheromone maps used in the HACO-SP and HACOH-SP are given by Table 10.11. The best results are indicated in bold.

Table 10.11: Runtimes for the SP on the QAP and MSSP Domains

SP	1D	2D	3D	Hybrid
QAP	**1.25**	2.54	2.65	1.75
MSSP	**2.70**	3.81	3.62	3.39

In contrast to the runtimes in Table 10.10, all of the times in Table 10.11 are at least a minute. This indicates that the selection perturbative hyper-heuristic is a more computationally intensive process than the selection constructive hyper-heuristic as both hyper-heuristics operate in the same domain.

In terms of the results, the usage of the 1D pheromone again produces the best runtimes of any of the pheromone map types in the QAP and MSSP domain. However this time, the worst-performing algorithms are the 2D HACO and 3D HACO respectively. The HACOH ran faster than the prior two algorithms in both domains. Part of this may be attributed to the usage of a combination of the prior map types as the usage of the 1D pheromone map in the hybrid can counterbalance the cost of the more computationally expensive 2D and 3D pheromone maps.

10.4.3 Generation Constructive Hyper-Heuristics

The runtimes for the different pheromone maps used in the HACO-GC and HACOH-GC are given by Table 10.12. The best results are indicated in bold.

Table 10.12: Runtimes for the GC on the 1BDPP and MSSP Domains

GC	1D	2D	3D	Hybrid
1BPP	**3.24**	6.22	5.18	6.88
MSSP	**0.04**	0.11	0.10	0.09

Table 10.12 highlights a stark difference between the choice of domain and its effect on runtimes. In particular, the 1BPP domain has the largest runtimes of any domain amongst the experiments with the fastest pheromone map type (1D) taking at least three minutes for a single run on aggregate over the benchmark.

However, the runtimes of the HACO-GC and HACOH-GC are very comparable to the HACO-SC and HACOH-SC in the MSSP domain, taking well under a minute regardless of the pheromone map type used in the hyper-heuristic.

While the use of the 1D pheromone map results in the best runtimes for hyper-heuristics in either domain, the use of the remaining maps in the hyper-heuristics results in runtimes of approximately twice as long. With the HACOH being the worst in the 1BPP domain, and the 2D HACO being the worst in the MSSP domain.

10.4.4 Generation Perturbative Hyper-Heuristics

The runtimes for the different pheromone maps used in the HACO-GP and HACOH-GP are given by Table 10.13. The best results are indicated in bold.

Table 10.13: Runtimes for the GP on the CVRP and MSSP Domains

GP	1D	2D	3D	Hybrid
CVRP	**0.16**	1.11	1.10	0.98
MSSP	**0.01**	0.12	0.17	0.14

The runtimes in Table 10.13 paint an interesting picture with regards to the effect of the different pheromone maps in the domains. The use of the 1D pheromone map results in a significantly faster runtime than the other types of maps which are around the 1 minute mark per run on average. This is similarly repeated in the MSSP domain where the use of the 1D pheromone map produces a runtime an order of magnitude faster than the other types of pheromone maps.

10.4.5 Analysis and Discussion

Due to the number of ways runtimes can be influenced, they are of tertiary importance in the wider context of this research. Nevertheless, they suggest some interesting trends with regards to how HACO and HACOH algorithms respond to the different problem domains and pheromone maps.

As the HACO and HACOH algorithms are modified versions of the AS algorithm, the majority of the computational considerations should be the same or similar to the baseline AS algorithm. That is, the HACO and HACOH algorithms do not meaningfully alter the operation of the AS except for how the AS algorithm interacts with the problem domain as it is shifted to the heuristic space instead of the solution space. There are computational considerations that might occur as a result of these modifications but they will be primarily determined by the specific problem domains and not the underlying algorithm. These would include solution construction and solution evaluation for instance.

However, that being said, there are several important points to consider concerning how the different pheromone maps affect the hyper-heuristics that make use of them.

The first point to consider is the effect of the 1D pheromone map. In all experiments, the use of the 1D pheromone map showed a consistently low runtime in comparison to all other types of pheromone maps. This held regardless of the domain or the type of hyper-heuristic.

This behaviour, the quick runtimes, is most definitely a product of the nature of the 1D pheromone map and its application. It has several benefits in terms of improving the runtimes of the algorithm. Namely, it has a much-reduced heuristic

space that it needs to search, having compressed the nominally 2D pheromone map into a single 1D representation. This is an easier structure to traverse and one that is less costly to traverse in general by taking up less memory than the other types of pheromone maps.

The other aspect behind the improved runtimes would then be the use of only a single ant. As in the FANT [5], the use of a single ant comes with several potential drawbacks but one of the major advantages is the reduced computational cost of using a single ant. The combination of the single ant on the reduced heuristic space, therefore, leads to major improvements in runtimes.

Therefore, the 1D pheromone map should be used in applications where speed is of paramount importance. This creates an ant-based hyper-heuristic with the ability to produce quick solutions, often in half the time than the other when using other types of pheromone maps. It strongly indicates that the 1D pheromone map has an advantage over the other pheromone maps when considering the relationship between the pheromone map and runtime for ant-based hyper-heuristics.

The next point to consider is the runtime performance of the hybrid. As the HACOH algorithm makes use of a list to determine the order of execution for each type of pheromone map in its overall execution, the nature of the list will be the primary determinant of the computational costs of running the HACOH algorithm. There were several different outcomes.

In the HACOH-SC, the hybrid was the worst-performing algorithm in terms of runtimes whereas, for the SP and GP, the hybrid was not the worst performer, with runtimes slower than the 1D pheromone map but faster than at least one of the other two remaining pheromone maps.

The final point to consider is the difference in runtimes between the uses of the 2D and 3D pheromone maps. Initially, it was assumed that, because of the increased size of the 3D pheromone map, using the 2D pheromone map in an ant-based hyper-heuristic would always perform faster than the 3D pheromone map. However, these results indicate that this is not the case.

For the HACO-SC, the use of 2D and 3D pheromone maps result in approximately identical runtimes and the difference is marginal for the HACO-GP as well. Only in the HACO-SP and HACO-GC (specifically in the 1BPP domain) are the differences more pronounced where the use of the 3D pheromone map takes longer in the SP-QAP domain and is faster in the other cases.

This suggests that the differences between the two pheromone maps are not as significant as originally hypothesized when it comes to the effect of the map on the runtime of the hyper-heuristic that makes use of it.

10.5 Pheromone Map Analysis

Any discussion of the results would be incomplete without talking about how the pheromone maps are used and modified by the ant-based hyper-heuristics. To that end, this section presents examples of each of the maps in all dimensions and for all of the hyper-heuristics, as a way of analysing the behaviour of the

different pheromone maps in relation to the hyper-heuristic. These maps were taken from the best runs from the main experiments presented in Section 10.2. Furthermore, the maps have been subject to a normalising transformation such that every pheromone value at cell (i, j) is converted to the range [0,1] where 0 represents no pheromone and 1 represents the maximum pheromone found on that map in terms of intensity. This allows for comparison in a scale neutral way as all pheromones will be normalised into the given range. All maps have been subject to this transformation with a key indicating the pheromone intensity provided for each one.

This is not an exhaustive presentation due to the number of maps generated in the course of this research but the examples, presented as heatmaps, should be illustrative of the wider point. More specifically, presenting a pheromone map to analyse the pheromone concentrations, is equivalent to plotting the evolution of a GA. In this way, the pheromone map provides a visual representation of the results of the execution of the ant-based hyper-heuristic as it will show where the ants travelled in the heuristic space as they searched it. The different types of maps also provide different levels of information, from general distributions in the 1D map to specific heuristic paths in the case of 3D maps and this section helps to clarify these differences.

Furthermore, the first few layers of the 3D pheromone map are presented to illustrate the progression of the pheromone through the layers; a full presentation of all of the layers would be both cumbersome and unnecessary for the wider point. Not all domains are represented. The reason for this is that the pheromone map itself is largely going to behave in similar ways across a different domain for the same type of hyper-heuristic so only a single domain is presented per hyper-heuristic.

Finally, the purpose of these heatmaps is to illustrate distributions of pheromone in the different dimensions. Sections 6.3, 7.6 and 8.5 provide the descriptions of the heuristics and components that make up the heuristic spaces represented on these heatmaps.

10.5.1 Selection Constructive Hyper-Heuristics

Figure 10.3 gives the two 1D pheromone maps produced by the HACO-SC and HACOH-SC for the QAP problem.

In terms of the distributions, Figure 10.3 indicates a stark difference in the normal HACO-SC as compared to the HACOH-SC. Whereas only one heuristic dominates in the case of the HACO-SC, several heuristics are highly prevalent in the HACOH-SC. As the HACOH-SC receives information from other types of pheromone maps, this would result in pheromone accumulations happening in other areas dictated by factors outside of the scope of the 1D pheromone map.

Figure 10.6 gives the 2D pheromone maps for the QAP domain. In contrast to the prior case, the 2D HACO-SC pheromone map has a broad dispersion of pheromone across its surface with only a few regions that stand out as predominant. Also, much more of the map has pheromone distributed across itself. By contrast, the 2D HACOH-SC pheromone map has a much sparser distribution of

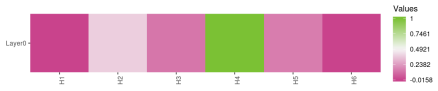

Fig. 10.1: HACO-SC

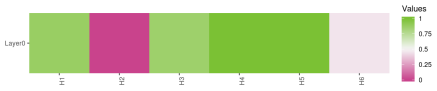

Fig. 10.2: HACOH-SC

Fig. 10.3: QAP 1D HACO Pheromone Maps

pheromone with only a handful of regions even accumulating some pheromone at the end.

Figure 10.10 provides the first three layers of a 3D HACO-SC's pheromone map. In this map, large segments of it are devoid of pheromone with clusters in each layer indicating where concentrations of pheromone accumulated during the algorithm's execution. In general, the pheromone concentration decreases from each layer and this is expected behaviour.

The heuristic space for the HACO-SC consists of heuristics and so as the layers progress, the number of heuristics that get selected will invariably decrease as some heuristics become favoured over others.

Figure 10.14 represents the first three layers of the 3D pheromone map of the HACOH-SC. In terms of these maps, the HACOH-SC behaves similarly to the HACOH-SC with an initial concentration of pheromone on the first layer that somewhat tapers off. Unlike the HACO-SC, there is more pheromone distributed in the map on the various layers and this reflects the influence of the other maps that share information.

The HACO-SC and HACOH-SC algorithms were incredibly close together in terms of their optimality and so it is not too surprising to see that the 3D versions of the algorithm have a similar pheromone distribution.

10.5.2 Selection Perturbative Hyper-Heuristics

Figure 10.17 gives the two 1D pheromone maps produced by the HACO-SC and HACOH-SC for the QAP problem.

In terms of the pheromone concentrations for the 1D pheromone maps for the HACO-SP and HACOH-SP in the MSSP domain, Figure 10.17, indicates a very similar trend to the prior HACO-SC and HACOH-SC 1D pheromone maps.

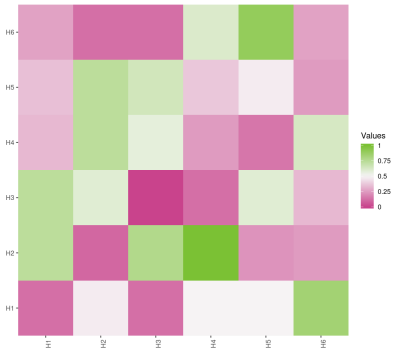

Fig. 10.4: HACO-SC

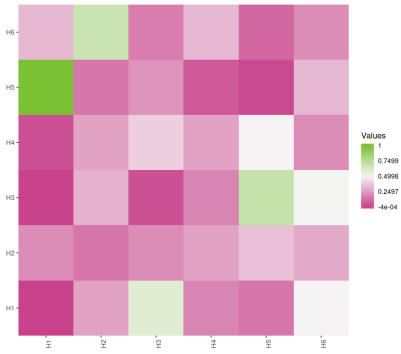

Fig. 10.5: HACOH-SC

Fig. 10.6: QAP 2D HACO Pheromone Maps

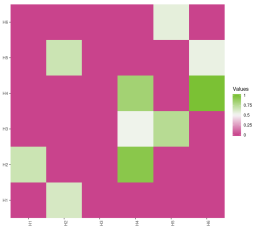

Fig. 10.7: Layer 0

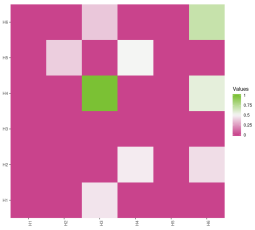

Fig. 10.8: Layer 1

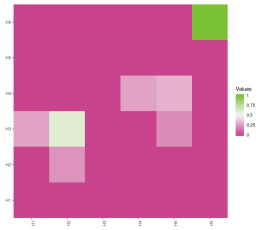

Fig. 10.9: Layer 2

Fig. 10.10: QAP 3D HACO Pheromone Map Layers

Specifically, in the case of the HACO-SP, only a single heuristic has emerged with most of the pheromone concentrating in that heuristic whereas the HACOH-SP has a much wider distribution of pheromone because it receives information from the other maps. As the 1D pheromone map cannot store much information about the path structure, this pheromone concentration serves to add additional heuristics to the selection potential whereas the 1D HACO-SP tends to fixate on a specific heuristic to the detriment of all others.

In Figure 10.20 both 2D pheromone maps of the HACO-SP and HACOH-SP for the MSSP domain are given. Similar to the comparison with the HACO-SC and HACOH-SC in the QAP domain, the 2D pheromone map for the HACO-SP shows a generally sparse map with a few regions of higher pheromone concentration. The 2D pheromone map for the HACOH-SP by contrast has a much wider distribution of pheromone that represents information from the other maps. Importantly, this distribution allows for the greater exploration of the space as

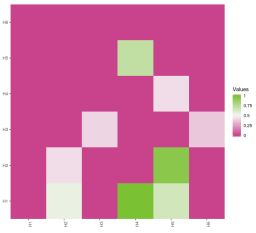

Fig. 10.11: Layer 0

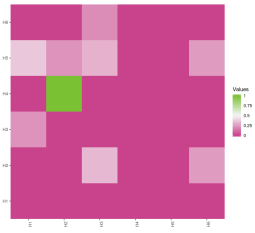

Fig. 10.12: Layer 1

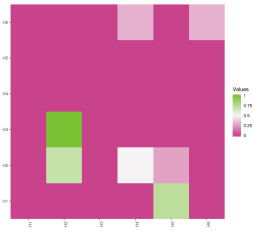

Fig. 10.13: Layer 2

Fig. 10.14: QAP 3D HACOH Pheromone Map Layers

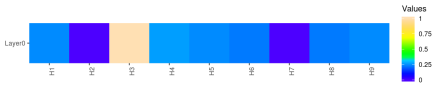

Fig. 10.15: HACO-SP

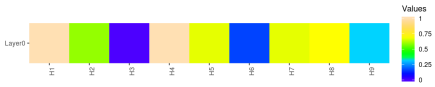

Fig. 10.16: HACOH-SP

Fig. 10.17: MSSP 1D HACO Pheromone Maps

more heuristics are likely to be included but to the detriment of exploitative potential.

In Figure 10.24, the first three layers of the HACO-SP in the MSSP are represented. Characteristically, most of the pheromone map is devoid of pheromone except for some specific regions. The first layer in particular has a very tight cluster of high pheromone regions in a row. This gives way to more dispersed clusters of pheromone in layers 1 and 2. By layer 2, most of the high pheromone areas are giving way to regions with much less pheromone.

Figure 10.28 represents the first three layers of the HACOH-SP's 3D pheromone map. In contrast to the HACO-SP, the layers of this pheromone map have their pheromone in a wider dispersal pattern and are generally of lower concentrations. There is more pheromone coverage, but this also means less concentration in specific areas.

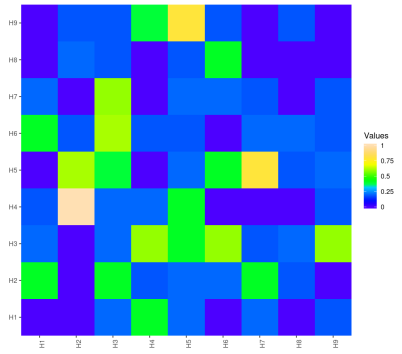

Fig. 10.18: HACO-SP

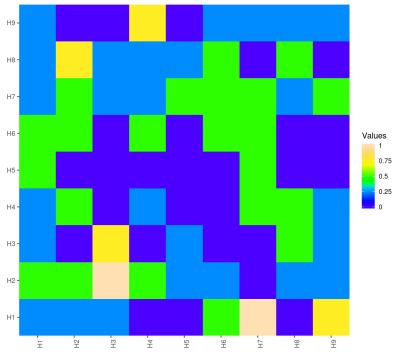

Fig. 10.19: HACOH-SP

Fig. 10.20: MSSP 2D HACO Pheromone Maps

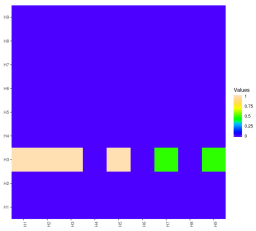

Fig. 10.21: Layer 0

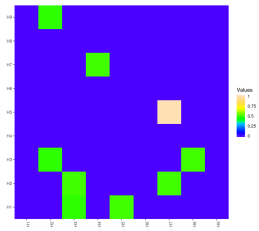

Fig. 10.22: Layer 1

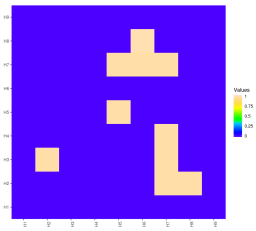

Fig. 10.23: Layer 2

Fig. 10.24: MSSP 3D HACO Pheromone Map Layers

10.5.3 Generation Constructive Hyper-Heuristics

Figure 10.31 gives the two 1D pheromone maps produced by the HACO-GC and HACOH-GC for the QAP problem.

The 1D pheromone maps for the HACO-GC and HACOH-GC are given in Figure 10.31 paints a different picture with regards to pheromone distribution than the prior selection hyper-heuristics. In the case of the HACO-GC, multiple components have large concentrations of pheromone whereas, in the case of HACOH-GC, the pheromone is distributed less broadly. The process of generating a constructive heuristic, as described previously, is certainly a difficult one and the 1D pheromone map has little ability to represent anything beyond the most basic information about how to assemble a heuristic and so, this distribution represents the distribution of a component's frequency in the heuristic generation.

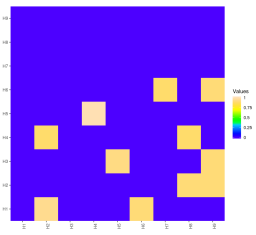

Fig. 10.25: Layer 0

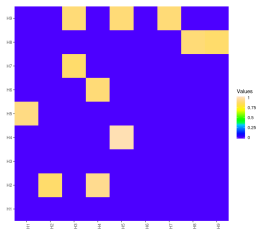

Fig. 10.26: Layer 1

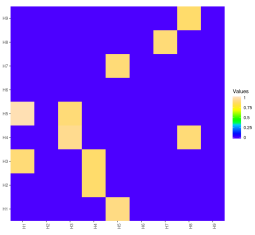

Fig. 10.27: Layer 2

Fig. 10.28: MSSP 3D HACOH Pheromone Map Layers

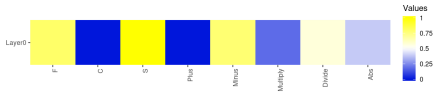

Fig. 10.29: HACO-GC

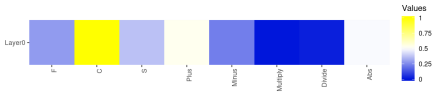

Fig. 10.30: HACOH-GC

Fig. 10.31: 1BPP 1D HACO Pheromone Maps

The HACO-GC is better able to assemble something more precisely and so has wider separations between components, whereas the HACOH-GC receives additional information from other sources that disrupts this fine-tuned process and so the pheromone distribution is broader.

This trend continues for the 2D pheromone maps for the HACO-GC and HACOH-GC given in Figure 10.34. The HACO-GC has a much more sparse distribution of pheromone in its 2D map which represents a better ability to capture the specific useful components whereas the HACO-H 2D pheromone map is much more broadly variable in its pheromone concentrations.

Figure 10.38 contains the first three layers of the HACO-GC for the 1BPP domain. This domain in particular brings the differences between pheromone maps into stark relief. Whereas the 1D and 2D pheromone maps in this section had some wider distribution of pheromone, pheromone in the 3D pheromone map is tightly clustered to specific areas. This is a strong indicator of the algorithm's

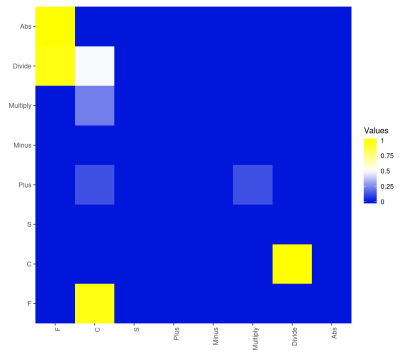

Fig. 10.32: HACO-GC

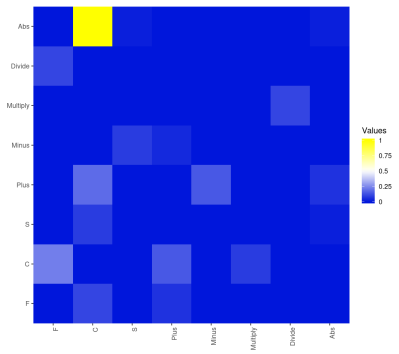

Fig. 10.33: HACOH-GC

Fig. 10.34: 1BPP 2D HACO Pheromone Maps

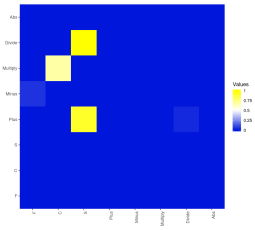

Fig. 10.35: Layer 0

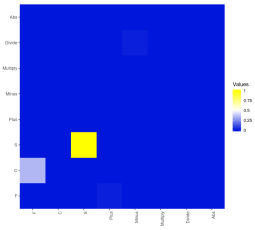

Fig. 10.36: Layer 1

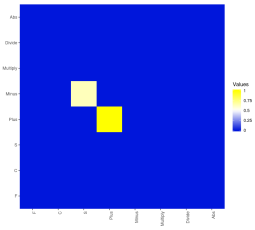

Fig. 10.37: Layer 2

Fig. 10.38: 1BPP 3D HACO Pheromone Map Layers

ability to leverage the information capacity of the 3D pheromone to precisely order the heuristic component selections into a few structured heuristics and is one of the reasons why the 3D HACO-GC did so well in this problem domain.

In Figure 10.42, three layers of the 3D pheromone map of the HACOH-GC are presented. The hybrid has a similar degree of pheromone concentration to the non-hybrid which does indicate to some degree that the hybrid algorithm was able to mitigate the information coming from the other pheromone maps and prevent the pheromone crowding that was observed in other maps. However, where the HACO-GC has relatively tightly clustered high pheromone regions, this is not the case for the HACOH-GC. Possibly the dispersal pattern reflects the influence of the other pheromone maps but this could also be the process of generating an alternative heuristic.

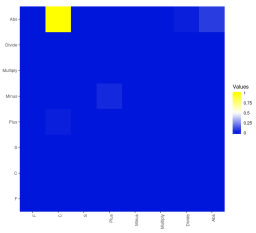

Fig. 10.39: Layer 0

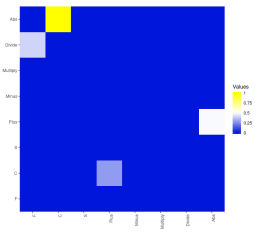

Fig. 10.40: Layer 1

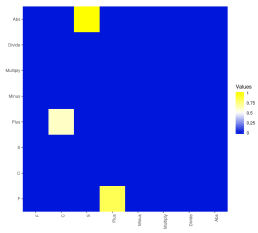

Fig. 10.41: Layer 2

Fig. 10.42: 1BPP 3D HACOH Pheromone Map Layers

10.5.3.1 Comparison of Generated Heuristics It can also be illustrative to examine some of the heuristics that are generated by the heuristic to see how they compare to when different pheromone maps are used in their creation. To that end, this section will discuss the best heuristics for a single instance, the first one in the 1BPP benchmark, to examine some of the differences between the different pheromone maps. The sheer number of heuristics makes the comparison of everyone impractical, but a few key examples are considered. These heuristics are presented in Appendix B.1.

One thing to note with regards to the comparisons is that the best heuristic does not necessarily completely map onto the pheromone perfectly. The best heuristic represents a snapshot of one potential state of the map during the algorithm's execution and the map can change as the search executes, especially as it explores the heuristic space such that it never quite gets to the exact state by the end of the execution. Nevertheless, examining some of the heuristics can be illustrative and this is discussed below.

With regards to the heuristic represented by Figure B.1, the most represented component is the domain attribute C. In contrast to the map, which showed C as being of low significance, with F and S being the most important attributes. In terms of the operators, Minus and divide were the most represented which does track with regards to the heuristic as those were the majority operators. Of course, the 1D pheromone map cannot provide more information other than the pheromone distribution so the heuristic does stand in isolation.

Figure B.2 shows a heuristic generated with a 2D pheromone map. In this regard, the heuristic has the S and F domain attributes as the most represented with the divide and absolute value operators being most prevalent. The heuristic is also smaller than the 1D pheromone map in terms of the number of operators.

Finally, Figure B.3 shows the heuristic generated with the 3D pheromone map. In this case, the heuristic has more diversity in terms of its components as all of the operators and domain attributes are represented but not in equal measure. This heuristic is similar to the one in Figure B.1 in terms of length but differs by having a more nested composition of components, which is especially

evident by the nesting of four operators at the beginning, something the other heuristics lack. Importantly, the multiply to divide link is represented on the first layer in the corresponding pheromone although the subsequent layers differ.

10.5.4 Generation Perturbative Hyper-Heuristics

Figure 10.45 gives the two 1D pheromone maps produced by the HACO-GP and HACOH-GP for the CVRP problem.

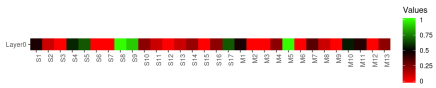

Fig. 10.43: HACO-GP

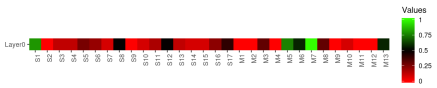

Fig. 10.44: HACOH-GP

Fig. 10.45: CVRP 1D HACO Pheromone Maps

Figure 10.45 represents the 1D pheromone maps for the HACO-GP and HACOH-GP for the CVRP domain. The trends observed for other 1D pheromone maps do not necessarily persist in this case. Specifically, the pheromone concentration in the HACO-GP shows two areas with large concentrations of pheromone whereas the HACOH-GP only has one region. The larger volume of components in this domain makes it a more difficult heuristic space to operate in but the 1D pheromone map was able to produce some concentrated regions.

Figure 10.48 provides the 2D pheromone maps for the HACO-GP and HACOH-GP. In terms of the distribution of pheromone, the HACO-GP has a much sparser distribution of pheromone in terms of the overall amount on the map, but this distribution is more spread out around the map. By contrast, the HACOH-GP has more pheromone around the map in general, but the largest areas of concentration are much closer together.

The first three layers of the HACO-GP in the CVRP domain are given in Figure 10.52. The large size of the pheromone map highlights the disparities in pheromone. Whereas the 2D pheromone maps for this problem were somewhat covered in pheromone, the concentrations in the layers of the 3D pheromone map are incredibly minimal. Outside of a few specific regions that represent

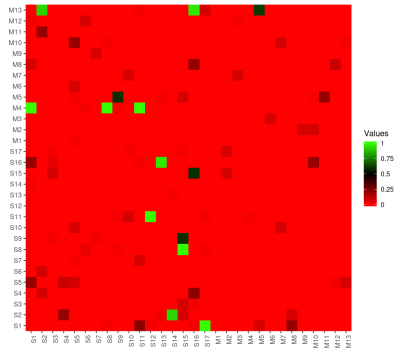

Fig. 10.46: HACO-GP

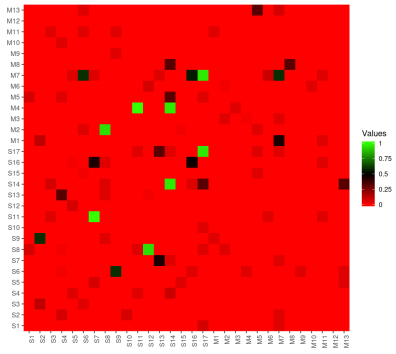

Fig. 10.47: HACOH-GP

Fig. 10.48: CVRP 2D HACO Pheromone Maps

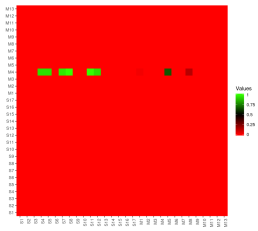

Fig. 10.49: Layer 0

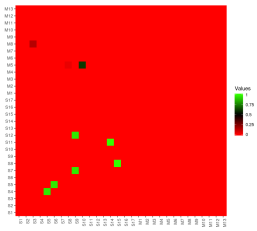

Fig. 10.50: Layer 1

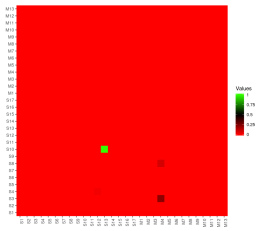

Fig. 10.51: Layer 2

Fig. 10.52: CVRP 3D HACO Pheromone Map Layers

high pheromone concentrations, most of the map layers are largely empty and the areas of high pheromone concentration are relatively tightly packed together.

By contrast, the pheromone concentrations for the first three layers of the 3D pheromone map of the HACOH-GP are much more widely dispersed throughout the layers, and as a consequence, there are far fewer areas of high pheromone concentration. This dispersal pattern is much more similar to the behaviour of a 2D pheromone map than the behaviour of a 3D pheromone map amongst the generation hyper-heuristics seen thus far.

10.5.4.1 Comparison of Generated Heuristics It can also be illustrative to examine some of the heuristics that are generated by the heuristic to see how they compare to when different pheromone maps are used in their creation. To that end, this section will discuss the best heuristics for a single instance, the first one in the 1BPP benchmark, to examine some of the differences between the

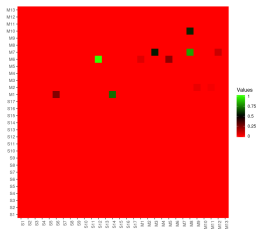

Fig. 10.53: Layer 0

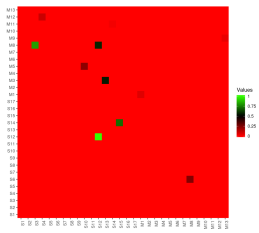

Fig. 10.54: Layer 1

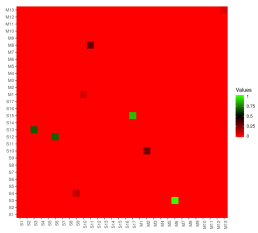

Fig. 10.55: Layer 2

Fig. 10.56: CVRP 3D HACOH Pheromone Map Layers

different pheromone maps. The sheer number of heuristics makes the comparison of everyone impractical, but a few key examples are considered. These heuristics are presented in Appendix B.2.

One thing to note with regards to the comparisons is that the best heuristic does not necessarily completely map onto the pheromone perfectly. The best heuristic represents a snapshot of one potential state of the map during the algorithm's execution and the map can change as the search executes, especially as it explores the heuristic space such that it never quite gets to the exact state by the end of the execution. Nevertheless, examining some of the heuristics can be illustrative and this is discussed below.

Figure B.4 provides a heuristic generated with the 1D pheromone map. The size of these heuristics is much larger than in the HACO-GC case but several factors influence that. Firstly, perturbative heuristics must manipulate an already created solution to function. This generally means they will need more operators and components to function properly than a heuristic that is used to create a solution from scratch as only meaningful changes to the solution will improve it.

With that in mind, M13 is the most represented mutator which does not necessarily track to the pheromone map. The selectors, S8 and S9, however, are quite prevalent in the heuristic.

Figure B.5 represents a heuristic constructed with a 2D pheromone map. Like the heuristic before it, this heuristic is also generally large although it only has 17 mutators as opposed to the 23 in the prior case. This repeats the trend observed in the prior HACO-GC comparison for the 2D pheromone map to result in heuristics that are smaller than the 1D pheromone. M13 is not as heavily represented both on the map and in the heuristic. Some of the linkages can be seen, like that between S1 and S17 but others are less explicit.

Figure B.6 gives the heuristic generated with a 3D pheromone map. This heuristic only has 18 operators in contrast to the prior two. So in general it seems as if the 2D pheromone map produces smaller heuristics than the 1D and 3D.

10.5.5 Analysis and Discussion

This section brings to light a number of the different capacities that the different types of pheromone maps are meant to represent. Firstly, between the 1D, 2D and 3D pheromone map, each map has a different information-carrying capacity and that limit is represented, at least partially, in how that map ends up concentrating pheromone.

For the non-hybrid maps, the 1D pheromone map has the propensity to concentrate pheromone into a few of the available regions of its already limited heuristic space. The 3D pheromone map by contrast can represent a long chain of information and thus, pheromone tends to be concentrated in a few areas per layer, with the totality of the concentrations representing the entire heuristic path.

The 2D pheromone strikes a balance between these two extremes with some ability to represent information and a smaller heuristic space and so these maps tend to produce specific clusters of high pheromone but also are not as empty as their 3D counterparts.

The type of hyper-heuristic does play a role in this behaviour. Pheromone maps used in generation hyper-heuristics, constructive or perturbative, have a greater tendency to form a few regions of high pheromone, which in these maps represents some optimal combination of heuristic components, than pheromone maps used in selective hyper-heuristics, constructive or perturbative. By contrast, these maps tend to retain more pheromone over a wider area in general.

The different nature of working with heuristic components over low-level heuristics does influence the way the algorithms distribute their pheromone on the maps. Choosing low-level heuristics is much less restrictive than choosing heuristic components and so that would naturally mean more of the heuristics can be chosen, leading to wider distributions of pheromone in the map.

The other aspect to consider is the effect of hybridisation. The HACOH algorithm operates by sharing information between three different pheromone maps (1D, 2D and 3D). This means that at any point in the algorithm's execution two pheromone maps are receiving pheromone update information from the other map currently being used. However as the maps are different in their construction, and in what heuristic space they are searching, there is a tendency in the algorithm to allow more pheromone to remain in every map than what is normally filtered out by a non-hybrid algorithm.

This capacity means that the algorithm will always struggle to produce the fine refinements that the other non-hybrid algorithms could produce in terms of their pheromone maps (cluster the pheromone on only the most important parts of the map) but it does provide a benefit in that the wider dispersion pattern of pheromone encourages exploration. As there is more pheromone around the maps made by the hybrid algorithm, better exploration can be facilitated through ants using that pheromone to explore more.

10.6 Comparison with Heuristics

While not the primary focus of this research, it is nevertheless important to contextualise the performance of the HACO and HACOH algorithms in a wider context to understand how well the hyper-heuristics function. In particular, the focus of this comparison is against existing heuristics. In the case of the generation hyper-heuristics, this is also an especially fair comparison as the generated heuristics can be compared against others in the domain. The focus, in this case, is to examine the differences between the hyper-heuristics and existing heuristics although this is not necessarily a primary objective of this research.

The values presented are based on the ratio calculation specified in Section 3.5. The F_R is used for all the domains as it makes the comparisons domain neutral and also simplifies the presentation across multiple heuristics. The best value will be indicated in bold. In addition to the F_R value, the standard deviation and rank of the method are presented. The rank simply indicates the ranked position of the method, heuristic or hyper-heuristic, when compared to all others with 1.00 indicating the best position. The rank evaluation is a simple and convenient metric for examining the different methods in relation to each other.

10.6.1 Selection Constructive Hyper-Heuristics

In terms of the results present in Table 10.14, the best performing methods are the HACO and HACO algorithms by a fairly large margin based on the F_R values. The best performing method, by a narrow margin, is the HACO making use of the 3D pheromone map although the 2D HACO is only 0.01 larger. Nevertheless, the gap between the construction heuristics and the hyper-heuristics is a significant enough gap to indicate the value of the hyper-heuristics. The heuristics for the QAP and MSSP domains are defined in Section 6.3.1.

Table 10.14: Comparison of SC Hyper-Heuristics with QAP Constructive Heuristics

SC-QAP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6
F_R	1.59	1.53	**1.52**	1.55	3.13	3.61	3.10	2.31	2.88	3.05
Std Dev	0.74	0.65	0.65	0.66	2.42	2.92	2.20	1.39	2.07	2.19
Rank	4.00	2.00	1.00	3.00	9.00	10.00	8.00	5.00	6.00	7.00

In Table 10.15, the best performing method is the HACO making use of the 1D pheromone map. The next best methods are the other HACO and HACOH algorithms, although a single construction heuristic (H8) can offer a competitive result in comparison to the hyper-heuristics. The remaining construction heuristics are all significantly worse than the best hyper-heuristic in terms of their F_R values.

Table 10.15: Comparison of SC Hyper-Heuristics with MSSP Constructive Heuristics

SC-MSSP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8
F_R	**1.187**	1.206	1.204	1.207	1.465	1.369	1.376	1.467	1.447	1.393	1.521	1.222
Std Dev	0.096	0.131	0.131	0.130	0.059	0.082	0.058	0.079	0.081	0.098	0.073	0.080
Rank	1.00	3.00	2.00	4.00	10.00	6.00	7.00	11.00	9.00	8.00	12.00	5.00

10.6.2 Selection Perturbative Hyper-Heuristics

The heuristics for the QAP and MSSP domains are defined in Section 6.3.2. These heuristics are used for comparison against the HACO and HACOH algorithms. Table 10.16 indicates that the best performing method in for SC-QAP domain, is the 2D HACO, followed closely by the 3D HACO and then the HACOH. The 1D HACO lags behind these methods but is still largely better than any of the extant perturbative heuristics. Only H5 and H6 come reasonably close to the performance of the hyper-heuristics in this case although the margin is still large.

Table 10.16: Comparison of SP Hyper-Heuristics with QAP Perturbative Heuristics

SP-QAP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11	H12
F_R	1.50	**1.37**	1.38	1.40	2.17	2.39	2.44	2.47	1.73	2.52	1.79	3.00	2.39	2.42	2.42	2.43
Std Dev	0.57	0.43	0.43	0.45	1.25	1.54	1.54	1.76	0.78	1.66	0.83	2.24	1.47	1.60	1.60	1.56
Rank	4.00	1.00	2.00	3.00	7.00	8.00	13.00	14.00	5.00	15.00	6.00	16.00	9.00	10.00	10.00	12.00

In Table 10.17, the best performing method, by a narrow margin, is the 3D HACO followed again, closely by the 2D HACO and HACOH. The 1D HACOH did the worst in this domain but still better than any of the existing perturbative heuristics. The difference between the heuristics and the hyper-heuristics is smaller in this domain than in Table 10.16 with H8, H9 and H1 coming closer to matching the performance of the hyper-heuristics but still being behind.

Table 10.17: Comparison of SP Hyper-Heuristics with MSSP Perturbative Heuristics

SP-MSSP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8	H9
F_R	1.114	1.092	**1.091**	1.094	1.195	1.212	1.224	1.212	1.207	1.208	1.201	1.178	1.178
Std Dev	0.049	0.057	0.056	0.056	0.038	0.072	0.089	0.069	0.065	0.073	0.077	0.047	0.047
Rank	4.000	2.000	1.000	3.000	7.000	12.000	13.000	11.000	9.000	10.000	8.000	5.000	5.000

10.6.3 Generation Constructive Hyper-Heuristics

A comparison of the construction heuristics and hyper-heuristics for the 1BPP domain is presented in Table 10.18. The 1BPP heuristics are defined in Section 7.8 and the MSSP heuristics are defined in Section 6.3.1.

This domain is the first domain where the hyper-heuristics (all four of them) are not predominant. Specifically, the best hyper-heuristic, the hybrid (HACOH-GC), is the second-best method behind H4 and H5 although the difference is very slight. The 1D HACO and 3D HACO algorithms lag significantly behind in performance at rank 9 and 7 respectively with the 2D HACO taking 4th place.

Table 10.18: Comparison of GC Hyper-Heuristics with 1BPP Constructive Heuristics

GC-1BPP	1D	2D	3D	Hybrid	H0	H1	H2	H3	H4	H5	H6
F_R	1.116	1.045	1.062	1.042	1.060	1.058	1.234	1.104	**1.037**	**1.037**	1.279
Std Dev	0.026	0.026	0.022	0.031	0.001	0.004	0.110	0.062	0.034	0.034	0.173
Rank	9.000	4.000	7.000	3.000	6.000	5.000	10.000	8.000	1.000	1.000	11.000

The 1BPP domain is, in the context of this research, the most expansive of the domains and in terms of scale, the most difficult. However, despite this, a hyper-heuristic was able to produce a solution that was almost as good as the best heuristics with a relatively small computational budget (750 iterations). Given the results seen in this research, it is well within reason that with a larger computational budget, one more in line with GP computational budgets, the performance of the algorithms would improve. However the same could be said for the 2D HACO which is behind the HACOH but not by a wide margin.

Table 10.19: Comparison of GC Hyper-Heuristics with MSSP Constructive Heuristics

GC-MSSP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8
F_R	1.058	**1.032**	1.038	1.037	1.465	1.369	1.376	1.467	1.447	1.393	1.521	1.222
Std Dev	0.022	0.021	0.020	0.020	0.059	0.082	0.058	0.079	0.081	0.098	0.073	0.080
Rank	4.00	1.00	3.00	2.00	10.00	6.00	7.00	11.00	9.00	8.00	12.00	5.00

The comparison of construction heuristics and hyper-heuristics in Table 10.19 indicates that the best-performing methods are all hyper-heuristics, with the 2D HACO showing the best performance with the 3D HACO and HACOH having a slightly worse but similar performance. The 1D HACO proves the worst in this situation but it is better by an order of magnitude in comparison to the

best heuristic, H8. This result mirrors the performance of the hyper-heuristics in other prior domains except for the 1BPP domain.

10.6.4 Generation Perturbative Hyper-Heuristics

A comparison of the construction heuristics and hyper-heuristics for the 1BPP domain is presented in Table 10.18. The CVRP heuristics are defined in Section 8.6 and the MSSP heuristics are defined in Section 6.3.2. Table 10.20 presents the comparison of perturbative heuristics and hyper-heuristics for the CVRP domain. Based on the results, the 1D HACO performed the best in this domain, with the H5 heuristic following as a relatively close second. The next best methods are the hyper-heuristics, which improve upon the heuristics by a wider margin. This is an unusual result because, in the other generation hyper-heuristic, the 2D HACO or HACOH performed better. The relative success of the 1D HACO over the 2D HACO and 3D HACO is therefore unusual for this type of problem.

Table 10.20: Comparison of GP Hyper-Heuristics with CVRP Perturbative Heuristics

GP-CVRP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8	H9
F_R	**1.1810**	1.2335	1.2378	1.2269	1.4815	1.4811	1.4427	1.4815	1.1821	1.4813	1.4815	1.4804	1.4436
Std Dev	0.10	0.14	0.14	0.13	0.14	0.14	0.14	0.14	0.06	0.14	0.14	0.14	0.14
Rank	1.00	4.00	5.00	3.00	12.00	9.00	6.00	11.00	2.00	10.00	12.00	8.00	7.00

In Table 10.21, the best performing method is the HACOH, followed by the 2D HACO, although the difference is relatively small. Despite this, the hyper-heuristics manage to outperform all of the heuristics by a relatively wide margin. The performance of the 2D HACO, 3D HACO and HACOH are much more closely aligned than the 1D HACO which is the reverse of the situation in the CVRP domain.

Table 10.21: Comparison of GP Hyper-Heuristics with MSSP Perturbative Heuristics

GP-MSSP	1D	2D	3D	Hybrid	H1	H2	H3	H4	H5	H6	H7	H8	H9
F_R	1.137	1.117	1.118	**1.116**	1.195	1.212	1.224	1.212	1.207	1.208	1.201	1.178	1.178
Std Dev	0.047	0.050	0.054	0.051	0.038	0.072	0.089	0.069	0.065	0.073	0.077	0.047	0.047
Rank	4.00	2.00	3.00	1.00	7.00	12.00	13.00	11.00	9.00	10.00	8.00	5.00	5.00

10.6.5 Comparison with Other Hyper-Heuristics

The purpose of this research is not to compare the ant-based hyper-heuristics with the state of the art. However, in terms of rounding out the comparison, as well as contextualising the performance of the hyper-heuristics, it can be interesting to compare the ant-based hyper-heuristic with another hyper-heuristic used in a similar domain.

There are, however, some issues with full comparison. Firstly, two of the problem domains (QAP and MSSP) have not been well studied in terms of hyper-heuristics so existing research is lacking. Secondly, the different methods and experimental conditions make a full comparison between them and these ant-based hyper-heuristics, difficult as the operating conditions are not the same.

To that end, this comparison will focus primarily on the BPP domain which has existing GP-based hyper-heuristics that can be readily compared to the HACO-GC.

Table 10.22: Comparison of HACO-GC with GP for 1BPP Datasets

Method	Uniform	Hard
1D HACO	1.1345	1.09785
2D HACO	1.0269	1.06405
3D HACO	1.0461	1.07769
HACOH	1.0201	1.06349
GP [106]	**1.0000**	**1.0004**

Table 10.22 provides a comparison between the HACO-GC algorithm (using different pheromone maps) and a GP-based hyper-heuristic operating in the 1BPP domain. The comparison presents the fitness ratios, F_R, calculated as per Equation 3.1, for the Uniform and Hard datasets in the 1BPP benchmark. The best result is indicated in bold.

In terms of the comparison, the GP method is the clear winner by a margin that grows or shrinks depending on the pheromone map used in the HACO-GC. The HACOH comes the closest in performance to the GP but the margin is still relatively large, even if the results only deviate from the best-known value by 2%. Ultimately a large part of this disparity can be attributed to the fact that GP is a mature technique whereas ant-based hyper-heuristics are much more recent. While they show promising results, especially in comparison to state-of-the-art, more work is needed to improve them.

10.6.6 Analysis and Discussion

Comparing the HACO and HACOH algorithms to existing heuristics is a good comparison to establish the relative capacities of the hyper-heuristics presented

in this research. Firstly, it establishes, in the case of generation hyper-heuristics, whether or not the hyper-heuristic can produce competitive heuristics to those that are already in use. Secondly, in the case of selection hyper-heuristics, it also determines how effective the existing heuristics can be combined to be more effective together than alone. However, in the context of this research, this type of comparison is also useful for examining where the different types of pheromone maps used in the ant-based hyper-heuristics are most effective.

In terms of the first point, the HACO and HACOH algorithms were very effective as generation constructive and generation perturbative hyper-heuristics, specifically in the MSSP and CVRP domains. In those domains, these algorithms produced the best results of any of the methods. The 1BPP domain proved slightly more difficult but at least one type of hyper-heuristic (HACOH) was competitive with the best heuristics.

In the case of the selection constructive and selection perturbative hyper-heuristics, across the QAP and MSSP domains, the hyper-heuristics were able to outperform the existing heuristics in all cases. The difference in performance is especially notable given that existing heuristics tend to produce solutions that are, in most cases, twice as worse as the solutions produced by the hyper-heuristics.

So in this sense, the ant-based hyper-heuristics can provide sufficiently good quality solutions in all the types of heuristics when compared to existing heuristics in each domain. However, this comparison has also reinforced the idea that the type of pheromone map used in the ant-based hyper-heuristic can significantly change its performance in relation to the others.

10.7 Summary

This chapter presented the results of the experiments conducted in this research. These results included a comparison of optimality, generality, runtimes, pheromone maps and comparisons with existing heuristics. The next chapter concludes the thesis which also includes a discussion of future work.

CHAPTER 11

Conclusion and Future Work

11.1 Introduction

This chapter presents the conclusions of this thesis as well as directions for expanding upon the current research in future. The rest of this chapter is organised as follows. Section 11.2 provides an overview of the thesis in its totality. Section 11.3 demonstrates how the research objectives were met. Section 11.4 details the major conclusions drawn from the research. Section 11.5 details possible research directions for future research. Finally, a summary of the chapter is given in Section 11.6.

11.2 An Overview of the Thesis

The research presented in this thesis is part of a wider initiative aimed at improving the use of ant algorithms in hyper-heuristics and extending the application of ant algorithms in hyper-heuristics. This thesis presented the HACO algorithm which is a general ant-based hyper-heuristic algorithm (HACO). This was employed for selection constructive, selection perturbative, generation constructive and generation perturbative hyper-heuristics, extending the general HACO algorithm to all four types of hyper-heuristics.

The thesis also provided a comprehensive study of how different pheromone maps, the principal operating mechanism of ant algorithms, could be used in these hyper-heuristics. This process entailed the use of three different pheromone map types (1D, 2D and 3D) in the four types of ant-based hyper-heuristics for different problem domains to empirically assess how each pheromone map type affected the performance of the underlying ant-based hyper-heuristic.

Furthermore, the research presented the HACOH algorithm which is a hybrid algorithm that makes use of three separate HACO algorithms, each with its distinct pheromone map. The goal of the HACOH algorithm is to test if the hybridisation of multiple distinct HACO algorithms (each with its distinct pheromone map) would perform better than non-hybridised HACO algorithms on their own.

To evaluate these algorithms, four different combinatorial optimisation problem domains were considered. The one-dimensional bin packing problem (1BPP), quadratic assignment problem (QAP), capacitated vehicle routing problem (CVRP) and the movie scene scheduling problem (MSSP). The first three are well-known

problem domains within the field of combinatorial optimisation and the last represents a new combinatorial problem to demonstrate the effectiveness of ant-based hyper-heuristics on a completely new problem domain.

A comprehensive analysis was performed on the results to ascertain the relative merits of the different pheromone maps in the different hyper-heuristics and problem domains. For most cases, it was found that there were distinct advantages for using specific types of pheromone maps for different problems and hyper-heuristics, with the algorithms that used 1D and 3D pheromone maps doing better than their counterparts. In other cases, the 2D pheromone map enabled the best hyper-heuristic performance. The HACOH algorithm did outperform the non-hybrid HACO algorithms in the GC-1BPP and GP-MSSP domains, signifying that there is some utility to be gained through the hybridisation process. The hybridisation process was found to have some improvements over some of the pheromone maps but not sufficiently to compete with the best type.

11.3 Thesis Objectives

There are several objectives put forward in Section 1.2 regarding the objectives that this research aimed to reach. This section details how the course of the research met these research objectives. The objectives are as follows.

11.3.1 Objectives One to Four

The first five objectives are as follows.

1. **To design and implement an ant-based selection constructive hyper-heuristic with 1D, 2D and 3D pheromone maps.**
2. **To design and implement an ant-based selection perturbative hyper-heuristic with 1D, 2D and 3D pheromone maps.**
3. **To design and implement an ant-based generation constructive hyper-heuristic with 1D, 2D and 3D pheromone maps.**
4. **To design and implement an ant-based generation perturbative hyper-heuristic with 1D, 2D and 3D pheromone maps.**

The first four objectives concern themselves with the development of an ant-based hyper-heuristic for one of the four types of hyper-heuristics. Throughout this research, four successful implementations of the HACO algorithm were developed and presented. These are the HACO-SC, HACO-SP, HACO-GC and HACO-GP. Each of these is capable of being used with one of the three pheromone maps (1D, 2D and 3D) that are also defined in this work. Therefore this work serves as a successful demonstration that hyper-heuristics can employ ant algorithms to drive their search in the heuristic space. These developments open the way for additional research to be done with ant-based hyper-heuristics as ant algorithms have not been used by generation hyper-heuristics until now.

Finally, there are two additional consequences of achieving this objective. The first is that ACO has been extended to generation hyper-heuristics which

is a novel application by itself. Secondly, however, the generality of the ACO technique itself has been extended in terms of its applicability to use in optimisation. One of the major limitations of the ACO has been its reliance on problems conforming to a graph-based representation. Problems that did not have this representation could not be solved in their native form without transformation. By moving the ACO to search a heuristic space, instead of a solution space, ACO can now be used by a hyper-heuristic to solve any problem that can be solved by heuristics. The approaches here can be extended to apply to any combinatorial optimisation problem for which the appropriate heuristic information exists or could be developed.

11.3.2 Objective Five

Objective Five is given as:

5. **To design and implement a hybridisation method such that all three pheromone maps can be used together in a hyper-heuristic.**

This objective concerns itself with the development of a hybridisation method to leverage the capacities of all three types of pheromone maps into a single algorithm. The HACOH algorithm has achieved this goal by incorporating three separate HACO algorithms (each with its distinct pheromone map) with a hybridisation framework to connect the HACO algorithms' search efforts. There are several implications of reaching this objective. Firstly, it shows that it is possible to leverage all three of the pheromone maps by a single algorithm.

Secondly, the HACOH algorithm also provides an alternative to the existing HACO algorithms by not requiring any selection of the pheromone map to use in the ant-based hyper-heuristic. As it makes use of all three, in their algorithms, it can provide an alternative to the non-hybrid HACO algorithms which are dependent on the choice of a single pheromone map.

11.3.3 Objective Six

The final objective is given as:

6. **To assess the effect of 1D, 2D and 3D pheromone maps have on the four types of ant-based hyper-heuristics as well as to compare the hybrid algorithm against the non-hybrid algorithms**

The sixth and final objective concerns itself with the empirical assessment of the effect of the different pheromone maps when used in the various hyper-heuristic (HACO and HAOCH) algorithms. It is the most important objective around which most of this research revolves. Based on the results of Chapter 10, several implications arise from meeting this objective.

Firstly, the results have confirmed that there is no free lunch regarding the choice of pheromone map for use in a HACO algorithm. The results demonstrated that different pheromone maps are meaningful in most of the experiments, but

that no singular map type is truly optimal for all hyper-heuristics or domains. There is a certain degree of granularity to this as well. For some domains like the SC-MSSP, the observed differences between the effects of the pheromone maps when used in the hyper-heuristics is relatively narrow. However, for others like the GC-1BPP, the margins are much more significant.

Secondly, the HACOH was not a unilateral success in terms of being able to outperform the non-hybrid HACO algorithms. Rather, it was only able to outperform the HACO algorithms in two domains. Notwithstanding the possibility of improving the hybridisation process, what these results indicate is that hybrid algorithms have something to offer ant-based hyper-heuristics without necessarily supplanting non-hybrid ant-based hyper-heuristics. The HACOH can be used in cases where choosing which pheromone map to use in the ant-based hyper-heuristic would be untenable, and it would be able to provide reasonable performance.

Thirdly, there are several implications from the results regarding the nature of the pheromone maps and their use in the ant-based hyper-heuristics. One of the hypotheses of this research was that the 3D pheromone map would be better suited for the generation hyper-heuristics as it has the information-carrying capacity to allow the hyper-heuristic to precisely refine (and therefore structure) its pheromone map to produce better heuristics. However, in practice, the 1D and 2D pheromone maps have proven more than sufficient for use in ant-based generation hyper-heuristics. Conversely, it was theorised that the 1D pheromone map would be preferable for the selection hyper-heuristics because of the stochastic nature of the low-level heuristics but in practice, 2D and 3D pheromone maps are capable of being used with great success in selection hyper-heuristics.

Finally, some implications follow from the above statements. The 1D pheromone map has faster performance and requires less computing resources, but it also has the least interpretability as a consequence of compressing the heuristic space into 1D. The 3D pheromone map is the opposite, offering high levels of interpretability but at a greater cost. The 2D pheromone map generally falls between the two although there are cases where this is not true. The performance of the maps in relation to the ant-based hyper-heuristic, however, is much more dependent on the type of hyper-heuristic and the problem domain.

This further exemplifies the fact that the choice of the optimal pheromone map for an ant-based hyper-heuristic is highly dependent on the nature of the problem domain, and which hyper-heuristic is being used. That is clear delineations of which type of pheromone map should be used for which ant-based hyper-heuristics are not easy determinations.

11.4 Conclusion

This research has demonstrated that the type of pheromone map used by an ant-based hyper-heuristic does have a significant impact on the performance of that ant-based hyper-heuristic across a variety of problem domains and hyper-heuristic types. Furthermore, in the course of performing this study, the applica-

tion of ant algorithms to hyper-heuristics was greatly increased by the successful application of ant algorithms to generation constructive and generation perturbative hyper-heuristics through the HACO-GC and HACO-GP algorithms.

The hybridisation method, HACOH, did not universally outperform its non-hybrid HACO counterparts in the experiments. Rather, what it demonstrated was that the hybridisation attempt was only able to partially outperform the non-hybrid algorithms in some of the domains, instead of all of them. The HACOH, therefore, does provide an alternative to needing to select a specific pheromone map for a given ant-based hyper-heuristic, but this choice trades off potentially much better performance. That is, there are domains where the HACOH algorithm would be the optimal choice but it does not predominate over the non-hybrid HACO algorithms.

Furthermore, the limitations of ACOs in terms of working on graph-based problems have been removed and generation hyper-heuristics have been extended by applying ACO to them where previously they had been ignored. One goal of future research, amongst others, is to expand on this foundation and begin to refine and optimise the algorithms to create the most optimal form of ant-based hyper-heuristics.

11.5 Future Work

Hyper-heuristics employing ant algorithms to drive the search of heuristic spaces is still largely unexplored in terms of hyper-heuristics research. While this research has been comprehensive, there are still several areas for additional research and room for improvement. These are discussed below.

- **Pheromone Map Selection:** One of the insights gained from this research is that the choice of using which pheromone map can significantly impact the performance of the underlying ant-based hyper-heuristic. However, there are less clear indications of when the different pheromone maps should be used for which hyper-heuristics or even problem domains. Hence there is a need to investigate the task of choosing the type of pheromone map when using ant-based hyper-heuristics as the problem of pheromone map selection, similar to the task of algorithm selection for other optimisation problems.
- **Alternative Ant Algorithms**: The ant algorithms used in this research have largely focused on the ant system and an adaptation of the fast ant system. There are many more ant algorithms that have expanded and developed upon the most basic types of ant algorithms. Investigating the applicability and effectiveness of ant colony algorithms like the MIN-MAX Ant System [108] for use in hyper-heuristics is an open question for future work.
- **Hybridising Multiple Ant-Based Hyper-Heuristics**: The hybridisation strategy used in this research focused on hybridising different HACO algorithms with different pheromone maps together. However, this leaves room for an ant algorithm that attempts to hybridise different types of ant-based hyper-heuristics into a hybrid ant colony hyper-heuristic. There is an existing body of work to demonstrate that hybridising hyper-heuristics can

be effective [2], therefore investigating the possibility of using hybrid ant-based hyper-heuristics remains an interesting direction for future research.
- **Algorithm Selection Optimisation**: One of the key points demonstrated by this research is that there can be niches for which specific HACO algorithms, and therefore pheromone map types, would be useful. Investigating algorithm selection algorithms, or algorithms that decide when to use other algorithms would be a good potential research direction for expanding the utility of ant-based hyper-heuristics.
- **Expanded Domain Application**: The HACO-GC and HACO-GP algorithms, are novel generation constructive and generation perturbative ant-based hyper-heuristics. Their potential for application in domains beyond those explored here remains an interesting possibility for future research given the relative paucity of ant-based generation hyper-heuristics and especially the lack of many generation perturbative hyper-heuristics in general.

11.6 Summary

This chapter has presented a comparison of different pheromone maps and their use and effect in the four types of hyper-heuristics as per the results of the study. A discussion of how the research objectives were achieved is provided as well. The study is concluded and a discussion of possible future directions for the research is given as well.

Bibliography

[1] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*. Springer US, 01 2003.

[2] N. Pillay and R. Qu, *Hyper-Heuristics: Theory and Applications*. Springer, 2018.

[3] J. Drake, A. Kheiri, E. Özcan, and E. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, 08 2019.

[4] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, 1992. [Online]. Available: https://ci.nii.ac.jp/naid/10000136323/en/

[5] E. Taillard, "Fant: Fast ant system." IDSIA, Via Cantonale 2C, 6928 Manno, Switzerland, Tech. Rep., 1998.

[6] S. Martello and P. Toth, *Bin-packing problem*. John Wiley and Sons, 1990.

[7] A. Schrijver and S.-V. (Berlin)., *Combinatorial Optimization: Polyhedra and Efficiency*, ser. Algorithms and Combinatorics. Springer, 2003, no. v. 1. [Online]. Available: https://books.google.co.za/books?id=mqGeSQ6dJycC

[8] B. Korte and J. Vygen, *Combinatorial Optimization Theory and algorithms*. Springer Berlin, 2019.

[9] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization," *ACM Computing Surveys*, vol. 35, no. 3, p. 268–308, 2003.

[10] E. Marais and W. D. Kok, *The soul of the white ant*. Osiran, 2009.

[11] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," *IEEE.*, vol. 2, p. 1477 Vol. 2, 02 1999.

[12] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric tsps by ant colonies," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 622–627.

[13] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-based search for combinatorial optimization: A critical survey," *Annals of Operations Research*, vol. 131, pp. 373–, 10 2004.

[14] Z. Abdmouleh, A. Gastli, L. Ben-Brahim, M. Haouari, and N. Al-Emadi, "Review of optimization techniques applied for the integration of distributed generation from renewable energy sources," *Renewable Energy*, vol. 113, 05 2017.

[15] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: Theory, Algorithms, and Applications*. Pearson Education Limited, 2014.

[16] M. Dorigo, T. Stützle, and N. Mastorakis, "An experimental study of the simple ant colony optimization algorithm," in *Proceedings of the WSES International Conference on Evolutionary Computation*, 2001, pp. 253–258.

[17] M. Dorigo and G. Di Caro, *The Ant Colony Optimization Meta-heuristic*. McGraw-Hill, 1999, pp. 11–32.

[18] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 2020.

[19] E. Taillard and L. Gambardella, "Adaptive memories for the quadratic assignment problem," IDSIA, Via Cantonale 2C, 6928 Manno, Switzerland, Tech. Rep., 1997.

[20] R. Qu and E. K. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems," *Journal of the Operational Research Society*, vol. 60, no. 9, p. 1273–1285, 2009.

[21] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the*

Operational Research Society, vol. 64, pp. 1695–1724, 07 2013.

[22] J. Drake, A. Kheiri, E. Özcan, and E. Burke, "Recent advances in selection hyper-heuristics," *European Journal of Operational Research*, vol. 285, pp. 405–428, 09 2020.

[23] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, p. 39–59, 1994.

[24] E. K. Burke, B. L. Maccarthy, S. Petrovic, and R. Qu, "Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning," *Practice and Theory of Automated Timetabling IV Lecture Notes in Computer Science*, p. 276–287, 2003.

[25] E. K. Burke, S. Petrovic, and R. Qu, "Case-based heuristic selection for timetabling problems," *Journal of Scheduling*, vol. 9, no. 2, p. 115–132, 2006.

[26] F. Glover and M. Laguna, *Tabu Search*. Boston, MA: Springer US, 1998, pp. 2093–2229. [Online]. Available: https://doi.org/10.1007/978-1-4613-0303-9_33

[27] P. Hansen and N. Mladenović, "Variable neighborhood search: Principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, p. 449–467, 2001.

[28] N. Pillay, "Evolving hyper-heuristics for the uncapacitated examination timetabling problem," *Journal of the Operational Research Society*, vol. 63, no. 1, p. 47–58, 2012.

[29] ——, "A study of evolutionary algorithm selection hyper-heuristics for the one-dimensional bin-packing problem," *South African Computer Journal*, vol. 48, 2012.

[30] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied*

Intelligence, vol. 37, no. 1, p. 1–11, 2011.

[31] L. N. Ahmed, E. Özcan, and A. Kheiri, "Solving high school timetabling problems worldwide using selection hyper-heuristics," *Expert Systems with Applications*, vol. 42, no. 13, p. 5463–5471, 2015.

[32] M. Lassouaoui, D. Boughaci, and B. Benhamou, "A hyper-heuristic method for max-sat," in *Proceedings of the International Conference on Metaheuristics and Nature Inspired Computer*, 2014, pp. 1–3.

[33] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem," *IEEE Congress on Evolutionary Computation*, 2010.

[34] J. Drake, "Crossover control in selection hyper-heuristics: Case studies using mkp and hyflex," Ph.D. dissertation, School of Computer Science, 2014.

[35] L. Han and G. Kendall, "Guided operators for a hyper-heuristic genetic algorithm," *Lecture Notes in Computer Science AI 2003: Advances in Artificial Intelligence*, p. 807–820, 2003.

[36] C. Rae and N. Pillay, "Investigation into an evolutionary algorithm hyper-heuristic for the nurse rostering problem," in *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling*, 08 2014, pp. 527–532.

[37] R. Raghavjee and N. Pillay, "A genetic algorithm selection perturbative hyper-heuristic for solving the school timetabling problem," *ORiON*, vol. 31, no. 1, p. 39, 2015.

[38] R. Aron, I. Chana, and A. Abraham, "A hyper-heuristic approach for resource provisioning-based scheduling in grid environment," *The Journal of Supercomputing*, vol. 71, no. 4, p. 1427–1450, 2015.

[39] J. Koza, "Genetic programming: On the programming of computers by means of natural selection," *Complex Adap. Syst.*, vol. 1, 01 1992.

[40] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, 01 2015.

[41] J. H. Drake, N. Kililis, and E. Özcan, "Generation of vns components with grammatical evolution for vehicle routing," *Lecture Notes in Computer Science Genetic Programming*, p. 25–36, 2013.

[42] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, no. 3, p. 205–219, 2009.

[43] M. Bader-El-Den and R. Poli, "Generating sat local-search heuristics using a gp hyper-heuristic framework," *Lecture Notes in Computer Science Artificial Evolution*, p. 37–49, 2008.

[44] A. S. Fukunaga, "Automated discovery of local search heuristics for satisfiability testing," *Evolutionary Computation*, vol. 16, no. 1, p. 31–61, 2008.

[45] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Grammatical evolution hyper-heuristic for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, p. 840–861, 2013.

[46] C. Contreras-Bolton and V. Parada, "Automatic design of algorithms for optimization problems," *2015 Latin America Congress on Computational Intelligence (LA-CCI)*, 2015.

[47] A. Cuesta-Cañada, L. Garrido, and H. Terashima-Marín, "Building hyper-heuristics through ant colony optimization for the 2d bin packing problem," *Lecture Notes in Computer Science Knowledge-Based Intelligent Information and Engineering Systems*, p. 654–660, 2005.

[48] A. S. Ferreira, A. Pozo, and R. A. Gonçalves, "An ant colony based hyper-heuristic approach for the set covering problem," *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 4, no. 1, p. 1–21, 2015.

[49] Z. Abd Aziz, "Ant colony hyper-heuristics for travelling salesman problem," *Procedia Computer Science*, vol. 76, pp. 534–538, 12 2015.

[50] P. Chen, G. Kendall, and G. Vanden Berghe, "An ant based hyper-heuristic for the travelling tournament problem," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Scheduling*, 2007, p. doi: 10.1109/SCIS.2007.367665.

[51] D. Beckedahl and N. Pillay, "A study of bi-space search for solving the one-dimensional bin packing problem," *Artificial Intelligence and Soft Computing Lecture Notes in Computer Science*, p. 277–289, 2020.

[52] F. Hruška and J. Kubalík, "Selection hyper-heuristic using a portfolio of derivative heuristics," *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 2015.

[53] D. Li, R. Zhan, D. Zheng, M. Li, and I. Kaku, "A hybrid evolutionary hyper-heuristic approach for intercell scheduling considering transportation capacity," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, p. 1072–1089, 2016.

[54] P. B. Miranda, R. B. Prudêncio, and G. L. Pappa, "H3ad: A hybrid hyper-heuristic for algorithm design," *Information Sciences*, vol. 414, pp. 340–354, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025516314323

[55] D. Johnson, "Near optimal bin packing algorithms," Ph.D. dissertation, Massachusetts Institute of Technology, 1967.

[56] R. Korf, "An improved algorithm for optimal bin packing." *IJCAI International Joint Conference on Artificial Intelligence*, pp. 1252–1258, 01 2003.

[57] E. Schreiber and R. Korf, "Improved bin completion for optimal bin packing and number partitioning," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 651–658, 08 2013.

[58] E. Burke, M. Hyde, G. Kendall, and J. Woodward, "Automating the packing heuristic design process with genetic programming," *Evolutionary computation*, vol. 20, pp. 63–89, 05 2011.

[59] E. Falkenauer and A. Delchambre, "A genetic algorithm for bin packing and line balancing," *Proceedings 1992 IEEE International Conference on Robotics and Automation*, 1992.

[60] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic activities," *Econometrica*, vol. 25, no. 1, p. 53, 1957.

[61] T. Dökeroğlu and A. Cosar, "A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem," *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 10–25, 06 2016.

[62] S. Adriaensen, G. Ochoa, and A. Nowé, "A benchmark set extension and comparative study for the hyflex framework," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 784–791.

[63] K. M. Anstreicher, "Recent advances in the solution of quadratic assignment problems," *Mathematical Programming*, vol. 97, no. 1, p. 27–42, 2003.

[64] Z. Drezner, P. Hahn, and e. Taillard, "Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods," *Annals of Operations Research*, vol. 139, pp. 65–94, 01 2005.

[65] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.

[66] C. Papadimitriou and K. Steiglitz, "Combinatorial optimization: Algorithms and complexity," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, 01 1982.

[67] M. Gendreau, G. Laporte, and Y. Potvin, "Metaheuristics for the capacitated vrp," *SIAM Monographs on Discrete Mathematics and Applications*,

01 2001.

[68] R. J. Marshall, M. Johnston, and M. Zhang, "Hyper-heuristics, grammatical evolution and the capacitated vehicle routing problem," *Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion - GECCO Comp 14*, 2014.

[69] K. Sim and E. Hart, "A combined generative and selective hyper-heuristic for the vehicle routing problem," *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO 16*, 2016.

[70] G. Mweshi, "A generation perturbative hyper-heuristic for combinatorial optimisation problems," Master's thesis, University of Pretoria, 2020.

[71] P. Gregory, A. Miller, and P. Prosser, "Solving the rehearsal problem with planning and with model checking," *European Conference on Artificial Intelligence*, vol. 16, 01 2004.

[72] N. Sakulsom and W. Tharmmaphornphilas, "Scheduling a music rehearsal problem with unequal music piece length," *Computers & Industrial Engineering*, vol. 70, 04 2014.

[73] T. C. E. Cheng, J. Diamond, and B. Lin, "Optimal scheduling in film production to minimize talent hold cost," *Journal of Optimization Theory and Applications*, vol. 79, pp. 479–492, 12 1993.

[74] A. Nordstrom and S. Tufekci, "A genetic algorithm for the talent scheduling problem," *Computers & Industrial Engineering*, vol. 21, pp. 927–, 10 1994.

[75] M. Garcia de la Banda, P. Stuckey, and G. Chu, "Solving talent scheduling with dynamic programming," *INFORMS Journal on Computing*, vol. 23, pp. 120–137, 02 2011.

[76] Y. Liu, Q. Sun, X. Zhang, and Y. Wu, "Research on the scheduling problem of movie scenes," *Discrete Dynamics in Nature and Society*, vol. 2019, pp. 1–8, 02 2019.

[77] X. Long and Z. Jinxing, "Scheduling problem of movie scenes based on three meta-heuristic algorithms," *IEEE Access*, vol. PP, pp. 1–1, 03 2020.

[78] E. Singh and N. Pillay, "Ant-based hyper-heuristics for the movie scene scheduling problem," *Proceedings 2021 ICAISC International Conference for Artificial Intelligence and Soft Computing*, confirm with prof 2021.

[79] S. Demeyer, "Research methods in computer science," *2011 27th IEEE International Conference on Software Maintenance (ICSM)*, 2011.

[80] B. J. Oates, "Researching information systems and computing," *SAGE*, 01 2005.

[81] C. Johnson, "What is research in computing science," *Computer Science Dept., Glasgow University. Electronic resource*, 2006. [Online]. Available: http://www.dcs.gla.ac.uk/johnson/teaching/research_skills/research.html

[82] R. L. Baskerville, "Investigating information systems with action research," *Communications of the Association for Information Systems*, vol. 2, 1999.

[83] G. I. Susman and R. D. Evered, "An assessment of the scientific merits of action research," *Administrative Science Quarterly*, vol. 23, no. 4, p. 582, 1978.

[84] T. Nyathi, "Automated design of genetic programming classification algorithms," Ph.D. dissertation, University of KwaZulu-Natal, 2019.

[85] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, p. 5–30, 1996.

[86] A. Scholl, R. Klein, and C. Jürgens, "Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem," *Computers & Operations Research*, vol. 24, no. 7, p. 627–645, 1997.

[87] J. E. Beasley, "Or-library." [Online]. Available: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/binpackinfo.html

[88] A. Scholl and R. Klein. [Online]. Available: http://www2.wiwi.uni-jena.de/Entscheidung/binpp/bin3dat.htm

[89] R. Burkard, E. Cela, S. Karisch, and F. Rendl, "Qaplib - a quadratic assignment problem library," 2018. [Online]. Available: http://anjos.mgi.polymtl.ca/qaplib/inst.html

[90] I. Xavier and E. Queiroga, "Cvrplib," 2014. [Online]. Available: http://vrp.atd-lab.inf.puc-rio.br/index.php/en/

[91] N. Pillay and R. Qu, "Assessing hyper-heuristic performance," *Journal of the Operational Research Society*, pp. 1–14, 2020.

[92] S. Babicki, D. Arndt, A. Marcu, Y. Liang, J. R. Grant, A. Maciejewski, and D. S. Wishart, "Heatmapper: web-enabled heat mapping for all," *Nucleic Acids Research*, vol. 44, no. W1, pp. W147–W153, 2016.

[93] K. Pearson, "Note on Regression and Inheritance in the Case of Two Parents," *Proceedings of the Royal Society of London Series I*, vol. 58, pp. 240–242, Jan. 1895.

[94] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[95] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

[96] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 03 1947.

[97] J. Dunn and O. J. Dunn, "Multiple comparisons among means," *American Statistical Association*, pp. 52–64, 1961.

[98] L. V. Hedges and I. Olkin, *Statistical methods for meta-analysis.* Academic Press, 2002.

[99] L. V. Hedges, "Distribution theory for glasss estimator of effect size and related estimators," *Journal of Educational Statistics*, vol. 6, no. 2, p. 107, 1981.

[100] J. Cohen, *Statistical power analysis for the behavioral sciences: Jacob Cohen.* Psychology Press, 2009.

[101] J. A. Durlak, "How to select, calculate, and interpret effect sizes," *Journal of Pediatric Psychology*, vol. 34, no. 9, p. 917–928, 2009.

[102] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[103] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.

[104] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT Press, 2010.

[105] V. Maniezzo and A. Colorni, "The ant system applied to the quadratic assignment problem," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 5, p. 769–778, 1999.

[106] E. Burke, M. Hyde, and G. Kendall, "Evolving bin packing heuristics with genetic programming," in *Parallel Problem Solving from Nature - PPSN IX*, 01 2006, pp. 860–869.

[107] G. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, vol. 12, no. 4, p. 568–581, 1964.

[108] T. Stutzle and H. Hoos, "Max-min ant system and local search for the traveling salesman problem," *Proceedings of 1997 IEEE International Confer-*

ence on Evolutionary Computation (ICEC 97), 1997.

Appendix A

Statistical Testing

This appendix contains the results of the statistical tests that are consulted in Section 10.2.

Table A.1: SC-QAP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
chr12a	45.5	0.00289	Reject H0	0.92640	154.5	0.96113	Do not Reject H0	0.64706	197	0.99979	Do not Reject H0	1.36489
chr12b	20	0.00007	Reject H0	1.65058	98	0.28070	Do not Reject H0	0.09284	208	0.99997	Do not Reject H0	1.61064
chr12c	15	0.00003	Reject H0	1.76567	70.5	0.04254	Do not Reject H0	0.64064	195	0.99971	Do not Reject H0	1.38419
chr15b	19	0.00006	Reject H0	1.73667	125	0.70513	Do not Reject H0	0.10012	208	0.99997	Do not Reject H0	1.87263
chr18b	217	0.99999	Do not Reject H0	3.12445	98.5	0.28773	Do not Reject H0	-0.05473	2	0.00000	Reject H0	3.64660
chr20a	33	0.00053	Reject H0	1.30710	99.5	0.30204	Do not Reject H0	0.13720	188	0.99919	Do not Reject H0	1.16509
chr22a	124.5	0.69796	Do not Reject H0	0.07592	189	0.99930	Do not Reject H0	1.37601	168	0.98991	Do not Reject H0	0.84839
els19	181	0.99790	Do not Reject H0	1.18598	85	0.13138	Do not Reject H0	0.40057	36	0.00081	Reject H0	1.43477
had12	42.5	0.00191	Reject H0	1.18798	122	0.66156	Do not Reject H0	0.01470	181.5	0.99809	Do not Reject H0	1.13567
had14	35	0.00068	Reject H0	1.37793	98	0.28022	Do not Reject H0	0.34602	181.5	0.99805	Do not Reject H0	1.14566
had16	17	0.00004	Reject H0	1.76483	142.5	0.89741	Do not Reject H0	0.60079	217	0.99999	Do not Reject H0	2.05219
had18	18	0.00005	Reject H0	1.93110	123	0.67612	Do not Reject H0	0.08157	214.5	0.99999	Do not Reject H0	2.12939
had20	40	0.00140	Reject H0	1.32997	113.5	0.52484	Do not Reject H0	0.04782	185	0.99878	Do not Reject H0	1.27698
kra32	209	0.99997	Do not Reject H0	1.62469	119	0.61422	Do not Reject H0	0.09060	18	0.00005	Reject H0	1.56437
sko49	33.5	0.00056	Reject H0	1.55873	126	0.71936	Do not Reject H0	0.12179	208.5	0.99997	Do not Reject H0	1.86860
sko56	41	0.00161	Reject H0	0.98992	132	0.79666	Do not Reject H0	0.22954	189.5	0.99935	Do not Reject H0	1.06701
sko64	35	0.00070	Reject H0	1.29390	96	0.25343	Do not Reject H0	0.07692	188	0.99919	Do not Reject H0	1.20954
sko72	41	0.00162	Reject H0	1.38835	78	0.07923	Do not Reject H0	0.39729	177.5	0.99671	Do not Reject H0	1.01436
sko81	34	0.00061	Reject H0	1.39011	91	0.19184	Do not Reject H0	0.36282	178	0.99692	Do not Reject H0	1.21106

Table A.2: SC-QAP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
chr12a	52	0.00638	Reject H0	0.92920	122.5	0.66853	Do not Reject H0	0.01131	157.5	0.97054	Do not Reject H0	0.66958
chr12b	32	0.00045	Reject H0	1.22369	170.5	0.99239	Do not Reject H0	0.91225	167.5	0.98935	Do not Reject H0	0.83331
chr12c	35	0.00070	Reject H0	1.36671	148.5	0.93509	Do not Reject H0	0.49452	115.5	0.55773	Do not Reject H0	0.08881
chr15b	23	0.00011	Reject H0	1.61119	135	0.82996	Do not Reject H0	0.29358	141.5	0.88947	Do not Reject H0	0.41994
chr18b	220	1.00000	Do not Reject H0	3.39784	140	0.87726	Do not Reject H0	0.32695	150	0.94251	Do not Reject H0	0.34669
chr20a	44	0.00240	Reject H0	0.95201	149	0.93757	Do not Reject H0	0.68815	135.5	0.83518	Do not Reject H0	0.46158
chr22a	126.5	0.72627	Do not Reject H0	0.12100	107.5	0.42595	Do not Reject H0	0.07988	195	0.99971	Do not Reject H0	1.44824
els19	191	0.99947	Do not Reject H0	1.54646	148	0.93231	Do not Reject H0	0.52290	117	0.58215	Do not Reject H0	0.08594
had12	52.5	0.00667	Reject H0	0.95650	130	0.77443	Do not Reject H0	0.35370	136.5	0.84637	Do not Reject H0	0.33418
had14	30	0.00033	Reject H0	1.45623	119	0.61477	Do not Reject H0	0.09072	91	0.19152	Do not Reject H0	0.30579
had16	14.5	0.00003	Reject H0	1.88173	95.5	0.24641	Do not Reject H0	0.19645	137	0.85042	Do not Reject H0	0.42753
had18	18	0.00005	Reject H0	1.89754	122.5	0.66862	Do not Reject H0	0.12491	128.5	0.75345	Do not Reject H0	0.22764
had20	37.5	0.00099	Reject H0	1.35207	104.5	0.37770	Do not Reject H0	0.18057	103.5	0.36203	Do not Reject H0	0.10688
kra32	205	0.99994	Do not Reject H0	1.54571	118.5	0.60629	Do not Reject H0	0.11050	119.5	0.62216	Do not Reject H0	0.02102
sko49	26	0.00018	Reject H0	1.55582	102	0.33912	Do not Reject H0	0.22065	112.5	0.50828	Do not Reject H0	0.16590
sko56	46	0.00309	Reject H0	0.79531	149	0.93757	Do not Reject H0	0.34271	149	0.93757	Do not Reject H0	0.50574
sko64	37	0.00093	Reject H0	1.23617	122.5	0.66844	Do not Reject H0	0.10917	106.5	0.40976	Do not Reject H0	0.01753
sko72	53.5	0.00761	Reject H0	0.89882	159.5	0.97562	Do not Reject H0	0.71404	119.5	0.62216	Do not Reject H0	0.22174
sko81	46	0.00309	Reject H0	1.07904	146	0.92079	Do not Reject H0	0.46125	128	0.74659	Do not Reject H0	0.13781

Table A.3: SC-QAP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
chr12a	9.563758389	0.02266266289	Reject H0
chr12b	20.15436242	0.0001576866564	Reject H0
chr12c	17.72	0.0005023774168	Reject H0
chr15a	0.76	0.8590085913	Do not Reject H0
chr15b	19.24	0.0002438703059	Reject H0
chr15c	4.68	0.1967857498	Do not Reject H0
chr18a	4.68	0.1967857498	Do not Reject H0
chr18b	23.16	3.74E-05	Reject H0
chr20a	11.4	0.009748366062	Reject H0
chr20b	2.718120805	0.4371566318	Do not Reject H0
chr20c	3.72	0.2933296062	Do not Reject H0
chr22a	14.91946309	0.001886782836	Reject H0
chr22b	6.76	0.07995381606	Do not Reject H0
chr25a	7.8	0.05033109786	Do not Reject H0
els19	10.44	0.01517348465	Reject H0
had12	12.02040816	0.007313576719	Reject H0
had14	18.18120805	0.000403573518	Reject H0
had16	17.31081081	0.0006099720579	Reject H0
had18	21.56375839	8.04E-05	Reject H0
had20	13.07432432	0.004478600818	Reject H0
kra30a	1.64	0.6503544781	Do not Reject H0
kra30b	2.44	0.4862320712	Do not Reject H0
kra32	20.36	0.000142940359	Reject H0
scr12	1.16	0.7626130659	Do not Reject H0
sko42	6.92	0.07449183186	Do not Reject H0
sko49	13.99328859	0.002914302398	Reject H0
sko56	15.16107383	0.001684047678	Reject H0
sko64	13.24	0.004145317909	Reject H0
sko72	10.76	0.01309702323	Reject H0
sko81	12.04	0.007247385935	Reject H0

Table A.4: SC-MSSP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
C_S_0_I0	11.81756757	0.008034954695	Reject H0
C_S_0_I1	9.75	0.02081525311	Reject H0
C_S_0_I2	13.89041096	0.003058181519	Reject H0
C_S_0_I3	7.630872483	0.05428943305	Do not Reject H0
C_S_0_I4	10.5704698	0.01429040106	Reject H0
C_S_1_I0	0.68	0.877897762	Do not Reject H0
C_S_1_I1	4.52	0.2105132556	Do not Reject H0
C_S_1_I2	8.04	0.0451922194	Reject H0
C_S_1_I3	12.2	0.00672852293	Reject H0
C_S_1_I4	3	0.3916251763	Do not Reject H0
C_S_2_I0	3.08	0.3794544638	Do not Reject H0
C_S_2_I1	1.8	0.6149349358	Do not Reject H0
C_S_2_I2	3.88	0.2747169155	Do not Reject H0
C_S_2_I3	21.24	9.39E-05	Reject H0
C_S_2_I4	6.92	0.07449183186	Do not Reject H0
C_S_3_I0	7.24	0.06462909894	Do not Reject H0
C_S_3_I1	5.16	0.160449136	Do not Reject H0
C_S_3_I2	12.36	0.006246400552	Reject H0
C_S_3_I3	7.16	0.06696915638	Do not Reject H0
C_S_3_I4	0.68	0.877897762	Do not Reject H0

Table A.5: SC-MSSP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	51.5	0.00600	Reject H0	1.10251	105	0.38548	Do not Reject H0	0.19061	176	0.99616	Do not Reject H0	1.01646
C_S_0_I1	67	0.03094	Do not Reject H0	0.79268	164	0.98456	Do not Reject H0	0.72827	189	0.99930	Do not Reject H0	1.30539
C_S_0_I2	38.5	0.00113	Reject H0	1.15372	104	0.36868	Do not Reject H0	0.01892	194	0.99967	Do not Reject H0	1.24606
C_S_0_I4	44	0.00237	Reject H0	1.10496	129.5	0.76644	Do not Reject H0	0.31144	191	0.99949	Do not Reject H0	1.27545
C_S_1_I2	51	0.00570	Reject H0	0.96097	147	0.92671	Do not Reject H0	0.60773	192	0.99955	Do not Reject H0	1.35638
C_S_1_I3	41	0.00161	Reject H0	1.33777	134	0.81925	Do not Reject H0	0.33049	196	0.99975	Do not Reject H0	1.61314
C_S_2_I3	210	0.99998	Do not Reject H0	2.31698	149	0.93757	Do not Reject H0	0.55424	18	0.00005	Reject H0	2.14116
C_S_3_I2	192	0.99955	Do not Reject H0	1.06181	142	0.89331	Do not Reject H0	0.53835	55	0.00903	Reject H0	0.93332

Table A.6: SC-MSSP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	37.5	0.00098	Reject H0	1.21510	94	0.22717	Do not Reject H0	0.22598	79	0.08465	Do not Reject H0	0.41677
C_S_0_I1	80.5	0.09563	Do not Reject H0	0.73446	146	0.92077	Do not Reject H0	0.04424	166.5	0.98815	Do not Reject H0	0.56116
C_S_0_I2	60	0.01545	Reject H0	0.93146	144.5	0.91197	Do not Reject H0	0.36439	142.5	0.89771	Do not Reject H0	0.43780
C_S_0_I4	62	0.01894	Do not Reject H0	0.93365	143	0.90130	Do not Reject H0	0.35836	159	0.97487	Do not Reject H0	0.69325
C_S_1_I2	47	0.00351	Reject H0	1.08860	89	0.17004	Do not Reject H0	0.30305	127	0.73309	Do not Reject H0	0.24813
C_S_1_I3	50	0.00506	Reject H0	1.04463	137	0.85012	Do not Reject H0	0.19091	156	0.96600	Do not Reject H0	0.46832
C_S_2_I3	210	0.99998	Do not Reject H0	2.36213	123	0.67590	Do not Reject H0	0.24219	157	0.96902	Do not Reject H0	0.74025
C_S_3_I2	187	0.99907	Do not Reject H0	1.01458	115	0.54951	Do not Reject H0	0.20902	136	0.84025	Do not Reject H0	0.30500

Table A.7: SP-QAP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
chr12a	18.76	0.0003064826283	Reject H0
chr12b	18.44	0.0003568663042	Reject H0
chr12c	5.16	0.160449136	Do not Reject H0
chr15a	20.28	0.000148506523	Reject H0
chr15b	14.2	0.002645179989	Reject H0
chr15c	20.68	0.0001226773353	Reject H0
chr18a	25.96	9.72E-06	Reject H0
chr18b	16.5704698	0.0008660483451	Reject H0
chr20a	17.48	0.0005629556077	Reject H0
chr20b	15.8	0.001246226177	Reject H0
chr20c	17	0.0007067423923	Reject H0
chr22a	15.08	0.001749544034	Reject H0
chr22b	26.6	7.14E-06	Reject H0
chr25a	21.8	7.18E-05	Reject H0
els19	26.76	6.61E-06	Reject H0
had12	13.60416667	0.003496595429	Reject H0
had14	14.31081081	0.002511210003	Reject H0
had16	16.42857143	0.0009261487597	Reject H0
had18	7.993288591	0.04615061412	Reject H0
had20	17.64	0.0005218132672	Reject H0
kra30a	22.44	5.28E-05	Reject H0
kra30b	22.84	4.36E-05	Reject H0
kra32	14.39597315	0.002412839803	Reject H0
scr12	16.2	0.00103178681	Reject H0
sko42	24.9527027	1.58E-05	Reject H0
sko49	21.24	9.39E-05	Reject H0
sko56	13.8	0.003190421908	Reject H0
sko64	19.64	0.0002015427626	Reject H0
sko72	19.64	0.0002015427626	Reject H0
sko81	15.56	0.001395557823	Reject H0

Table A.8: SP-QAP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
chr12a	23	0.00011	Reject H0	1.60134	103.5	0.36219	Do not Reject H0	0.07270	197	0.99979	Do not Reject H0	1.49884
chr12b	21	0.00008	Reject H0	1.55834	114	0.53306	Do not Reject H0	0.16778	214	0.99999	Do not Reject H0	1.67723
chr15a	23	0.00011	Reject H0	1.49558	145	0.91447	Do not Reject H0	0.31880	210	0.99998	Do not Reject H0	1.74540
chr15b	28	0.00025	Reject H0	1.56409	115	0.54951	Do not Reject H0	0.06434	199	0.99985	Do not Reject H0	1.58638
chr15c	23	0.00011	Reject H0	1.58917	97	0.26691	Do not Reject H0	0.07949	200	0.99987	Do not Reject H0	1.51617
chr18a	3	0.00000	Reject H0	2.90705	138	0.85958	Do not Reject H0	0.48899	223	1.00000	Do not Reject H0	2.87499
chr18b	20.5	0.00007	Reject H0	1.61101	109	0.45044	Do not Reject H0	0.28516	199	0.99985	Do not Reject H0	1.49221
chr20a	23	0.00011	Reject H0	1.75457	70	0.04075	Do not Reject H0	0.73499	189	0.99930	Do not Reject H0	1.38446
chr20b	11	0.00001	Reject H0	1.93854	103.5	0.36218	Do not Reject H0	0.18197	205.5	0.99995	Do not Reject H0	1.79024
chr20c	12	0.00002	Reject H0	2.20362	101	0.32410	Do not Reject H0	0.10339	209	0.99997	Do not Reject H0	1.89052
chr22a	13	0.00002	Reject H0	1.88995	87	0.14983	Do not Reject H0	0.35751	209	0.99997	Do not Reject H0	1.72496
chr22b	25	0.00015	Reject H0	1.60172	117.5	0.59024	Do not Reject H0	0.05833	201	0.99989	Do not Reject H0	1.62370
chr25a	15	0.00003	Reject H0	1.89618	74	0.05749	Do not Reject H0	0.43632	207	0.99996	Do not Reject H0	1.66635
els19	15	0.00003	Reject H0	1.94612	152	0.95145	Do not Reject H0	0.67043	222	1.00000	Do not Reject H0	2.89010
had12	34	0.00058	Reject H0	1.36735	131	0.78678	Do not Reject H0	0.28660	200	0.99988	Do not Reject H0	1.56661
had14	20	0.00006	Reject H0	1.59286	112	0.50000	Do not Reject H0	0.08952	196.5	0.99977	Do not Reject H0	1.40844
had16	26	0.00018	Reject H0	1.76280	109.5	0.45859	Do not Reject H0	0.00000	199.5	0.99986	Do not Reject H0	1.69339
had18	46.5	0.00325	Reject H0	1.17841	119.5	0.62233	Do not Reject H0	0.07141	177	0.99651	Do not Reject H0	1.15976
had20	15.5	0.00003	Reject H0	1.84449	132.5	0.80261	Do not Reject H0	0.23118	212	0.99998	Do not Reject H0	2.23335
kra30a	5	0.00000	Reject H0	2.36715	111	0.48346	Do not Reject H0	0.17095	219	1.00000	Do not Reject H0	2.12990
kra30b	20	0.00007	Reject H0	1.71201	74	0.05749	Do not Reject H0	0.53867	200.5	0.99988	Do not Reject H0	1.46079
kra32	31.5	0.00041	Reject H0	1.48644	109	0.45023	Do not Reject H0	0.05298	191	0.99950	Do not Reject H0	1.38284
scr12	15	0.00003	Reject H0	1.85578	67	0.03096	Do not Reject H0	0.63560	195	0.99971	Do not Reject H0	1.34604
sko42	10	0.00001	Reject H0	2.39416	92.5	0.20913	Do not Reject H0	0.12557	205.5	0.99995	Do not Reject H0	1.93291
sko49	26	0.00018	Reject H0	1.62154	113.5	0.52481	Do not Reject H0	0.03605	202.5	0.99991	Do not Reject H0	1.81979
sko56	41	0.00162	Reject H0	0.96544	153.5	0.95744	Do not Reject H0	0.49279	193	0.99961	Do not Reject H0	1.25382
sko64	26	0.00018	Reject H0	1.33700	150.5	0.94490	Do not Reject H0	0.47980	203	0.99992	Do not Reject H0	1.93387
sko72	19.5	0.00006	Reject H0	1.71796	130.5	0.77861	Do not Reject H0	0.30553	215	0.99999	Do not Reject H0	2.15387
sko81	24	0.00013	Reject H0	1.63788	99.5	0.30202	Do not Reject H0	0.10844	195.5	0.99973	Do not Reject H0	1.50084

Table A.9: SP-QAP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
chr12a	50	0.00506	Reject H0	0.98004	173	0.99430	Do not Reject H0	0.94621	163	0.98280	Do not Reject H0	0.82002
chr12b	26	0.00018	Reject H0	1.41298	132	0.79661	Do not Reject H0	0.13967	139.5	0.87302	Do not Reject H0	0.29671
chr15a	23	0.00011	Reject H0	1.56410	103	0.35446	Do not Reject H0	0.04102	134	0.81925	Do not Reject H0	0.30327
chr15b	46	0.00309	Reject H0	1.14722	139	0.86862	Do not Reject H0	0.44190	140	0.87726	Do not Reject H0	0.41971
chr15c	32	0.00045	Reject H0	1.20417	165	0.98604	Do not Reject H0	0.71149	151	0.94713	Do not Reject H0	0.59190
chr18a	10	0.00001	Reject H0	2.18433	131	0.78468	Do not Reject H0	0.23190	152	0.95145	Do not Reject H0	0.58956
chr18b	46	0.00309	Reject H0	1.02393	177	0.99653	Do not Reject H0	1.04175	171	0.99283	Do not Reject H0	0.84732
chr20a	39	0.00123	Reject H0	1.27873	153.5	0.95742	Do not Reject H0	0.65008	109	0.45049	Do not Reject H0	0.03053
chr20b	41	0.00162	Reject H0	1.23003	168	0.98991	Do not Reject H0	0.88339	157	0.96902	Do not Reject H0	0.70115
chr20c	19	0.00006	Reject H0	1.87138	143	0.90075	Do not Reject H0	0.52463	131	0.78468	Do not Reject H0	0.31477
chr22a	22	0.00009	Reject H0	1.58806	138	0.85963	Do not Reject H0	0.43151	122	0.66088	Do not Reject H0	0.12255
chr22b	44	0.00240	Reject H0	1.01451	165	0.98607	Do not Reject H0	0.73982	165	0.98606	Do not Reject H0	0.73086
chr25a	22	0.00009	Reject H0	1.49898	146.5	0.92381	Do not Reject H0	0.42619	123.5	0.68334	Do not Reject H0	0.03358
els19	36	0.00081	Reject H0	1.22074	155	0.96275	Do not Reject H0	0.51177	192	0.99955	Do not Reject H0	1.12750
had12	39	0.00119	Reject H0	1.26920	116.5	0.57466	Do not Reject H0	0.07336	133	0.80976	Do not Reject H0	0.34323
had14	41.5	0.00170	Reject H0	1.17241	167.5	0.98958	Do not Reject H0	0.91228	146	0.92118	Do not Reject H0	0.56835
had16	53	0.00714	Reject H0	0.98155	162	0.98108	Do not Reject H0	0.76904	158	0.97205	Do not Reject H0	0.73564
had18	52	0.00637	Reject H0	1.05490	121.5	0.65348	Do not Reject H0	0.08413	126	0.71941	Do not Reject H0	0.14085
had20	24	0.00013	Reject H0	1.46902	144	0.90816	Do not Reject H0	0.42320	164	0.98457	Do not Reject H0	0.69890
kra30a	2.5	0.00000	Reject H0	2.31157	120	0.62999	Do not Reject H0	0.00603	107	0.41785	Do not Reject H0	0.17324
kra30b	42	0.00185	Reject H0	1.14360	174.5	0.99524	Do not Reject H0	0.99183	152	0.95145	Do not Reject H0	0.54815
kra32	54.5	0.00843	Reject H0	0.90824	142	0.89359	Do not Reject H0	0.23710	140	0.87797	Do not Reject H0	0.26226
scr12	29	0.00029	Reject H0	1.42876	120.5	0.63781	Do not Reject H0	0.05898	96	0.25346	Do not Reject H0	0.40321
sko42	33.5	0.00056	Reject H0	1.47790	172.5	0.99397	Do not Reject H0	0.87681	154.5	0.96108	Do not Reject H0	0.59031
sko49	37	0.00093	Reject H0	1.28315	136.5	0.84529	Do not Reject H0	0.35770	137.5	0.85493	Do not Reject H0	0.36862
sko56	55	0.00903	Reject H0	0.79459	138	0.85966	Do not Reject H0	0.02789	157	0.96904	Do not Reject H0	0.36078
sko64	24.5	0.00014	Reject H0	1.58065	95.5	0.24682	Do not Reject H0	0.15775	132.5	0.80245	Do not Reject H0	0.35032
sko72	32	0.00045	Reject H0	1.37124	125.5	0.71229	Do not Reject H0	0.14163	144	0.90779	Do not Reject H0	0.41750
sko81	24	0.00013	Reject H0	1.61609	111.5	0.49173	Do not Reject H0	0.07772	102	0.33912	Do not Reject H0	0.17580

Table A.10: SP-MSSP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
C_S_0_I0	21.20134228	9.56E-05	Reject H0
C_S_0_I1	11.20945946	0.01064552744	Reject H0
C_S_0_I2	14.52517986	0.002270843076	Reject H0
C_S_0_I3	27.99280576	3.64E-06	Reject H0
C_S_0_I4	27.87755102	3.85E-06	Reject H0
C_S_1_I0	25.48	1.23E-05	Reject H0
C_S_1_I1	28.68	2.61E-06	Reject H0
C_S_1_I2	12.36	0.006246400552	Reject H0
C_S_1_I3	13.56	0.003569571998	Reject H0
C_S_1_I4	24.76	1.73E-05	Reject H0
C_S_2_I0	11.72	0.008406539411	Reject H0
C_S_2_I1	17	0.0007067423923	Reject H0
C_S_2_I2	13.5	0.003671131797	Reject H0
C_S_2_I3	19.4	0.000225970331	Reject H0
C_S_2_I4	17.72	0.0005023774168	Reject H0
C_S_3_I0	27.96	3.70E-06	Reject H0
C_S_3_I1	25.96	9.72E-06	Reject H0
C_S_3_I2	1.16	0.7626130659	Do not Reject H0
C_S_3_I3	22.76	4.53E-05	Reject H0
C_S_3_I4	29	2.24E-06	Reject H0

Table A.11: SP-MSSP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	15	0.00003	Reject H0	1.98793	100	0.30829	Do not Reject H0	0.10833	207.5	0.99996	Do not Reject H0	1.88576
C_S_0_I1	29	0.00028	Reject H0	1.56342	90	0.18000	Do not Reject H0	0.21149	189.5	0.99935	Do not Reject H0	1.40832
C_S_0_I2	26.5	0.00016	Reject H0	1.33175	92	0.19305	Do not Reject H0	0.23698	194.5	0.99972	Do not Reject H0	1.28970
C_S_0_I3	14	0.00002	Reject H0	2.55567	148.5	0.94109	Do not Reject H0	0.48795	215	0.99999	Do not Reject H0	2.74337
C_S_0_I4	0	0.00000	Reject H0	3.26659	78.5	0.08180	Do not Reject H0	0.44556	224.5	1.00000	Do not Reject H0	2.74070
C_S_1_I0	10	0.00001	Reject H0	2.05633	143	0.90075	Do not Reject H0	0.38574	215	0.99999	Do not Reject H0	2.26744
C_S_1_I1	2	0.00000	Reject H0	2.24505	113.5	0.52481	Do not Reject H0	0.05150	225	1.00000	Do not Reject H0	2.48111
C_S_1_I2	32	0.00045	Reject H0	1.54033	104	0.37000	Do not Reject H0	0.07924	193	0.99961	Do not Reject H0	1.51966
C_S_1_I3	49	0.00448	Reject H0	1.24787	149	0.93759	Do not Reject H0	0.45497	189	0.99930	Do not Reject H0	1.55783
C_S_1_I4	12	0.00002	Reject H0	2.08075	158	0.97180	Do not Reject H0	0.59858	214	0.99999	Do not Reject H0	2.45268
C_S_2_I0	33	0.00053	Reject H0	1.43461	95	0.24037	Do not Reject H0	0.26929	185	0.99877	Do not Reject H0	1.21057
C_S_2_I1	29	0.00029	Reject H0	1.46402	130	0.77235	Do not Reject H0	0.16493	201	0.99989	Do not Reject H0	1.59879
C_S_2_I2	29.5	0.00031	Reject H0	1.61097	99.5	0.30186	Do not Reject H0	0.15582	194	0.99967	Do not Reject H0	1.53483
C_S_2_I3	32	0.00045	Reject H0	1.21512	123	0.67592	Do not Reject H0	0.47463	191	0.99948	Do not Reject H0	1.28222
C_S_2_I4	34	0.00061	Reject H0	1.45263	134	0.81925	Do not Reject H0	0.33008	193	0.99961	Do not Reject H0	1.66369
C_S_3_I0	5	0.00000	Reject H0	2.54364	116	0.56589	Do not Reject H0	0.01133	215	0.99999	Do not Reject H0	2.15626
C_S_3_I1	9	0.00001	Reject H0	2.74341	102	0.33915	Do not Reject H0	0.28722	214	0.99999	Do not Reject H0	2.64415
C_S_3_I3	12	0.00002	Reject H0	2.04894	105	0.38578	Do not Reject H0	0.30636	214	0.99999	Do not Reject H0	2.44156
C_S_3_I4	2	0.00000	Reject H0	2.49552	71	0.04430	Do not Reject H0	0.54093	219	1.00000	Do not Reject H0	2.23568

Table A.12: SP-MSSP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	20	0.00007	Reject H0	1.73075	109.5	0.45853	Do not Reject H0	0.00019	104.5	0.37743	Do not Reject H0	0.08081
C_S_0_I1	36	0.00080	Reject H0	1.30945	127.5	0.74042	Do not Reject H0	0.19299	111.5	0.49171	Do not Reject H0	0.00720
C_S_0_I2	37.5	0.00090	Reject H0	1.24527	146	0.92728	Do not Reject H0	0.43821	132	0.80210	Do not Reject H0	0.21256
C_S_0_I3	20.5	0.00006	Reject H0	1.87218	138.5	0.87214	Do not Reject H0	0.51791	165	0.98795	Do not Reject H0	0.83360
C_S_0_I4	1	0.00000	Reject H0	2.88770	136	0.84136	Do not Reject H0	0.33714	101.5	0.33120	Do not Reject H0	0.11178
C_S_1_I0	11	0.00001	Reject H0	1.82076	98	0.28072	Do not Reject H0	0.31756	127	0.73309	Do not Reject H0	0.01836
C_S_1_I1	0	0.00000	Reject H0	2.73232	151	0.94713	Do not Reject H0	0.54093	144	0.90779	Do not Reject H0	0.55630
C_S_1_I2	35	0.00070	Reject H0	1.24875	144	0.90782	Do not Reject H0	0.23983	136.5	0.84526	Do not Reject H0	0.18073
C_S_1_I3	56	0.01010	Reject H0	1.01608	136	0.84027	Do not Reject H0	0.21926	160	0.97676	Do not Reject H0	0.59635
C_S_1_I4	19	0.00006	Reject H0	1.66645	153	0.95549	Do not Reject H0	0.70501	191	0.99947	Do not Reject H0	1.29188
C_S_2_I0	42	0.00185	Reject H0	1.07764	148	0.93231	Do not Reject H0	0.44446	127	0.73309	Do not Reject H0	0.16955
C_S_2_I1	32	0.00045	Reject H0	1.32476	134	0.81925	Do not Reject H0	0.31965	148	0.93231	Do not Reject H0	0.53262
C_S_2_I2	31.5	0.00040	Reject H0	1.58395	112.5	0.50835	Do not Reject H0	0.00813	100.5	0.31602	Do not Reject H0	0.15177
C_S_2_I3	58	0.01244	Reject H0	0.64956	166	0.98754	Do not Reject H0	0.49032	173	0.99435	Do not Reject H0	0.74067
C_S_2_I4	42	0.00185	Reject H0	1.37039	117	0.58215	Do not Reject H0	0.12096	134.5	0.82468	Do not Reject H0	0.44813
C_S_3_I0	23	0.00011	Reject H0	1.61806	180	0.99760	Do not Reject H0	0.63465	157	0.96902	Do not Reject H0	0.52303
C_S_3_I1	9	0.00001	Reject H0	2.75507	130	0.77247	Do not Reject H0	0.27884	112.5	0.50828	Do not Reject H0	0.03711
C_S_3_I3	8	0.00001	Reject H0	2.32700	102	0.33913	Do not Reject H0	0.04096	81	0.09923	Do not Reject H0	0.42043
C_S_3_I4	10	0.00001	Reject H0	1.86472	119	0.61426	Do not Reject H0	0.07110	92.5	0.20926	Do not Reject H0	0.44801

Table A.13: GC-1BPP Friedman Test Results (u120)

	Friedman Test		
Instance	χ^2	P Value	Outcome
u120_00	29.7890625	1.53E-06	Reject H0
u120_01	33.30215827	2.78E-07	Reject H0
u120_02	31.08461538	8.16E-07	Reject H0
u120_03	38.44615385	2.27E-08	Reject H0
u120_04	33.67391304	2.32E-07	Reject H0
u120_05	26.38636364	7.92E-06	Reject H0
u120_06	20.80147059	0.00011575807	Reject H0
u120_07	35.15217391	1.13E-07	Reject H0
u120_08	29.4	1.85E-06	Reject H0
u120_09	23.63414634	2.98E-05	Reject H0
u120_10	33.24390244	2.86E-07	Reject H0
u120_11	26.84782609	6.34E-06	Reject H0
u120_12	36.58695652	5.63E-08	Reject H0
u120_13	30.38059701	1.15E-06	Reject H0
u120_14	32.32835821	4.46E-07	Reject H0
u120_15	29.67768595	1.61E-06	Reject H0
u120_16	35.55	9.32E-08	Reject H0
u120_17	22.45238095	5.25E-05	Reject H0
u120_18	31.58955224	6.39E-07	Reject H0
u120_19	35	1.22E-07	Reject H0

Table A.14: GC-1BPP Friedman Test Results (u250)

	Friedman Test		
Instance	χ^2	P Value	Outcome
u250_00	31.22142857	7.64E-07	Reject H0
u250_01	22.64028777	4.80E-05	Reject H0
u250_02	28.08088235	3.49E-06	Reject H0
u250_03	25.09285714	1.48E-05	Reject H0
u250_04	28.13868613	3.40E-06	Reject H0
u250_05	27.48175182	4.67E-06	Reject H0
u250_06	32.86861314	3.43E-07	Reject H0
u250_07	25.2919708	1.34E-05	Reject H0
u250_08	31.20967742	7.68E-07	Reject H0
u250_09	28.87769784	2.38E-06	Reject H0
u250_10	29.32867133	1.91E-06	Reject H0
u250_11	25.10526316	1.47E-05	Reject H0
u250_12	18.97058824	0.0002772540178	Reject H0
u250_13	28.06666667	3.52E-06	Reject H0
u250_14	29.56551724	1.70E-06	Reject H0
u250_15	24.94285714	1.59E-05	Reject H0
u250_16	24.3	2.16E-05	Reject H0
u250_17	27.59558824	4.42E-06	Reject H0
u250_18	33.4379562	2.60E-07	Reject H0
u250_19	33.64285714	2.36E-07	Reject H0

Table A.15: GC-1BPP Friedman Test Results (u500)

	Friedman Test		
Instance	χ^2	P Value	Outcome
u500_00	27.28767123	5.12E-06	Reject H0
u500_01	32.31428571	4.49E-07	Reject H0
u500_02	37	4.60E-08	Reject H0
u500_03	30.20833333	1.25E-06	Reject H0
u500_04	34.93006993	1.26E-07	Reject H0
u500_05	30.23239437	1.23E-06	Reject H0
u500_06	33.19014085	2.94E-07	Reject H0
u500_07	31.35	7.17E-07	Reject H0
u500_08	32.5	4.11E-07	Reject H0
u500_09	29.14583333	2.09E-06	Reject H0
u500_10	27.53424658	4.55E-06	Reject H0
u500_11	34.65	1.44E-07	Reject H0
u500_12	35.97857143	7.57E-08	Reject H0
u500_13	21.27464789	9.23E-05	Reject H0
u500_14	29.34246575	1.90E-06	Reject H0
u500_15	25.60416667	1.15E-05	Reject H0
u500_16	21.35036496	8.90E-05	Reject H0
u500_17	35.89655172	7.88E-08	Reject H0
u500_18	34.39583333	1.63E-07	Reject H0
u500_19	35.15172414	1.13E-07	Reject H0

Table A.16: GC-1BPP Friedman Test Results (u1000 & HARD)

	Friedman Test		
Instance	χ^2	P Value	Outcome
u1000_00	24.52447552	1.94E-05	Reject H0
u1000_01	28.11724138	3.43E-06	Reject H0
u1000_02	24.65413534	1.82E-05	Reject H0
u1000_03	34.15862069	1.83E-07	Reject H0
u1000_04	16.07913669	0.001092396967	Reject H0
u1000_05	30.26174497	1.22E-06	Reject H0
u1000_06	30.97241379	8.61E-07	Reject H0
u1000_07	36.53191489	5.78E-08	Reject H0
u1000_08	34.13286713	1.86E-07	Reject H0
u1000_09	32.05479452	5.10E-07	Reject H0
u1000_10	22.24647887	5.80E-05	Reject H0
u1000_11	24.45	2.01E-05	Reject H0
u1000_12	31.84137931	5.65E-07	Reject H0
u1000_13	22.76223776	4.53E-05	Reject H0
u1000_14	34.14285714	1.85E-07	Reject H0
u1000_15	32.25	4.64E-07	Reject H0
u1000_16	31.43571429	6.88E-07	Reject H0
u1000_17	29.52739726	1.73E-06	Reject H0
u1000_18	35.52413793	9.44E-08	Reject H0
u1000_19	32.93835616	3.32E-07	Reject H0
HARD0	16.975	0.0007151588344	Reject H0
HARD1	30.61363636	1.03E-06	Reject H0
HARD2	19.44736842	0.0002209263387	Reject H0
HARD3	26.57480315	7.23E-06	Reject H0
HARD4	22.30434783	5.64E-05	Reject H0
HARD5	33.13740458	3.01E-07	Reject H0
HARD6	10.84615385	0.01258754317	Reject H0
HARD7	19.91803279	0.0001765126392	Reject H0
HARD8	23.70247934	2.88E-05	Reject H0
HARD9	24.87401575	1.64E-05	Reject H0

Table A.17: GC-1BPP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
u120_00	15	0.000004	Reject H0	1.97136	15	0.000004	Reject H0	1.72659	180	0.99787	Do not Reject H0	1.18679
u120_01	8	0.000006	Reject H0	2.98235	53	0.006032	Reject H0	0.96597	197.5	0.99983	Do not Reject H0	1.54245
u120_02	11	0.000006	Reject H0	2.38211	35.5	0.000301	Reject H0	1.48096	192.5	0.99965	Do not Reject H0	1.52648
u120_03	0	0.000001	Reject H0	4.51974	56	0.005693	Reject H0	0.79322	219	1.00000	Do not Reject H0	2.60857
u120_04	0	0.000001	Reject H0	4.51807	67.5	0.028319	Do not Reject H0	0.77241	224	1.00000	Do not Reject H0	3.25968
u120_05	16.5	0.000026	Reject H0	2.08401	99	0.283988	Do not Reject H0	0.16363	203	0.99994	Do not Reject H0	1.86800
u120_06	28	0.000177	Reject H0	1.74838	54	0.006266	Reject H0	0.64849	181.5	0.99829	Do not Reject H0	1.30657
u120_07	2.5	0.000001	Reject H0	3.00855	25	0.000072	Reject H0	1.52294	208	0.99997	Do not Reject H0	1.75888
u120_08	3	0.000001	Reject H0	2.73192	22	0.000037	Reject H0	1.94640	173.5	0.99517	Do not Reject H0	1.13120
u120_09	23.5	0.000068	Reject H0	2.16984	66.5	0.021377	Do not Reject H0	0.64412	190.5	0.99953	Do not Reject H0	1.91449
u120_10	11.5	0.000006	Reject H0	2.13674	55.5	0.004599	Reject H0	1.07086	197	0.99985	Do not Reject H0	1.49862
u120_11	33	0.000310	Reject H0	1.81386	39	0.000978	Reject H0	0.94054	182.5	0.99870	Do not Reject H0	1.03390
u120_12	1	0.000001	Reject H0	3.86206	29.5	0.000188	Reject H0	1.55695	216	1.00000	Do not Reject H0	2.14488
u120_13	12	0.000010	Reject H0	2.91901	56	0.007967	Reject H0	0.86374	200.5	0.99991	Do not Reject H0	1.74198
u120_14	18	0.000037	Reject H0	2.28609	40	0.001067	Reject H0	1.28792	189.5	0.99946	Do not Reject H0	1.61303
u120_15	18	0.000016	Reject H0	2.91647	45.5	0.001324	Reject H0	1.08697	191	0.99958	Do not Reject H0	1.63002
u120_16	1.5	0.000001	Reject H0	4.19914	41.5	0.001292	Reject H0	1.18042	214.5	0.99999	Do not Reject H0	2.74305
u120_17	27	0.000103	Reject H0	1.78054	37.5	0.000565	Reject H0	0.98787	170.5	0.99376	Do not Reject H0	1.14948
u120_18	14.5	0.000015	Reject H0	2.38840	54.5	0.005934	Reject H0	0.81324	201	0.99991	Do not Reject H0	1.85638
u120_19	1.5	0.000001	Reject H0	3.43355	43.5	0.001768	Reject H0	1.08789	211	0.99998	Do not Reject H0	2.01129
u250_00	6	0.000005	Reject H0	2.63473	43.5	0.001962	Reject H0	0.89959	213	0.99999	Do not Reject H0	2.28479
u250_01	35.5	0.000601	Reject H0	1.59106	12	0.000010	Reject H0	1.91757	145	0.91636	Do not Reject H0	1.11812
u250_02	11.5	0.000011	Reject H0	3.24108	48	0.003278	Reject H0	1.03175	208.5	0.99997	Do not Reject H0	2.58201
u250_03	13	0.000015	Reject H0	2.11854	99.5	0.293948	Do not Reject H0	0.07646	212.5	0.99999	Do not Reject H0	2.49876
u250_04	17.5	0.000037	Reject H0	2.69455	47.5	0.003056	Reject H0	1.12064	200	0.99988	Do not Reject H0	2.17466
u250_05	33	0.000476	Reject H0	1.87098	59	0.012948	Reject H0	0.97287	176	0.99614	Do not Reject H0	1.13926
u250_06	7.5	0.000006	Reject H0	2.49334	49	0.003676	Reject H0	1.05580	201	0.99990	Do not Reject H0	1.93911
u250_07	14	0.000011	Reject H0	3.25696	51	0.003342	Reject H0	0.95965	204.5	0.99996	Do not Reject H0	2.18250
u250_08	5.5	0.000004	Reject H0	3.14794	90.5	0.177514	Do not Reject H0	0.41532	204.5	0.99995	Do not Reject H0	2.10898
u250_09	7	0.000005	Reject H0	2.46134	66.5	0.023665	Do not Reject H0	0.59848	204	0.99994	Do not Reject H0	1.88082
u250_10	9	0.000006	Reject H0	2.25248	9	0.000007	Reject H0	2.03679	187	0.99919	Do not Reject H0	1.48675
u250_11	12	0.000010	Reject H0	2.73927	45	0.001821	Reject H0	1.09514	197.5	0.99982	Do not Reject H0	1.63067
u250_12	20	0.000051	Reject H0	2.00646	67	0.026375	Do not Reject H0	0.53586	186	0.99902	Do not Reject H0	1.76172
u250_13	24	0.000093	Reject H0	2.28266	72	0.035764	Do not Reject H0	0.58124	193	0.99967	Do not Reject H0	1.85608
u250_14	22.5	0.000083	Reject H0	1.76517	63.5	0.020512	Do not Reject H0	0.47124	193	0.99965	Do not Reject H0	1.55576
u250_15	12	0.000006	Reject H0	2.78062	38.5	0.000825	Reject H0	1.12413	195.5	0.99980	Do not Reject H0	1.77250
u250_16	17	0.000023	Reject H0	2.02864	78	0.064962	Do not Reject H0	0.26365	204.5	0.99995	Do not Reject H0	1.90183
u250_17	19	0.000041	Reject H0	2.08077	48	0.002688	Reject H0	0.67121	190	0.99949	Do not Reject H0	1.65513
u250_18	5	0.000004	Reject H0	2.78455	56.5	0.008868	Reject H0	0.49298	213.5	0.99999	Do not Reject H0	2.36282
u250_19	3	0.000003	Reject H0	3.10204	51.5	0.005264	Reject H0	1.01102	206.5	0.99996	Do not Reject H0	2.23179

Table A.18: GC-1BPP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
u500_00	10	0.000010	Reject H0	2.128618	72.5	0.048291	Do not Reject H0	0.527127	200.5	0.999883	Do not Reject H0	1.789599
u500_01	11.5	0.000012	Reject H0	3.516355	66.5	0.024046	Do not Reject H0	0.728428	211.5	0.999983	Do not Reject H0	3.225706
u500_02	4.5	0.000003	Reject H0	2.680490	53	0.005521	Reject H0	0.591677	213	0.999987	Do not Reject H0	2.260154
u500_03	7	0.000004	Reject H0	2.684319	15	0.000022	Reject H0	1.900843	184.5	0.998849	Do not Reject H0	2.008999
u500_04	10.5	0.000011	Reject H0	2.374916	83.5	0.115654	Do not Reject H0	0.090179	221.5	0.999998	Do not Reject H0	2.992402
u500_05	3.5	0.000003	Reject H0	3.678997	91	0.187686	Do not Reject H0	0.301829	220	0.999997	Do not Reject H0	3.314998
u500_06	15.5	0.000023	Reject H0	2.161327	21	0.000055	Reject H0	1.216354	190.5	0.999480	Do not Reject H0	1.507833
u500_07	10.5	0.000010	Reject H0	3.020784	62	0.018234	Do not Reject H0	0.687582	208.5	0.999974	Do not Reject H0	2.430497
u500_08	17	0.000036	Reject H0	1.856970	43	0.001829	Reject H0	0.655224	187	0.999118	Do not Reject H0	1.568740
u500_09	11	0.000013	Reject H0	2.006001	72.5	0.048425	Do not Reject H0	0.383256	204	0.999935	Do not Reject H0	1.752855
u500_10	16	0.000030	Reject H0	2.103211	31	0.000342	Reject H0	1.317486	180	0.997705	Do not Reject H0	1.649755
u500_11	0.5	0.000002	Reject H0	2.716142	71	0.038551	Do not Reject H0	0.836732	209	0.999973	Do not Reject H0	2.023776
u500_12	10.5	0.000010	Reject H0	3.780688	50	0.004390	Reject H0	0.846917	211.5	0.999983	Do not Reject H0	2.825608
u500_13	7.5	0.000006	Reject H0	1.876911	51	0.005160	Reject H0	0.884540	193.5	0.999654	Do not Reject H0	1.494700
u500_14	12.5	0.000016	Reject H0	2.286906	54.5	0.007796	Reject H0	0.382386	203	0.999931	Do not Reject H0	2.253067
u500_15	15.5	0.000027	Reject H0	2.533421	95.5	0.245117	Do not Reject H0	0.064465	210	0.999979	Do not Reject H0	2.564090
u500_16	22.5	0.000072	Reject H0	2.067152	38.5	0.000980	Reject H0	0.778627	177.5	0.996923	Do not Reject H0	1.690178
u500_17	3	0.000002	Reject H0	2.688082	22	0.000077	Reject H0	0.964551	213.5	0.999989	Do not Reject H0	2.060397
u500_18	12	0.000013	Reject H0	2.663884	37.5	0.000882	Reject H0	1.122056	198	0.999840	Do not Reject H0	2.161339
u500_19	0	0.000001	Reject H0	3.604593	53	0.006239	Reject H0	0.920516	217.5	0.999994	Do not Reject H0	2.793224
u1000_00	12	0.000015	Reject H0	2.197582	91.5	0.193505	Do not Reject H0	0.148209	207.5	0.999965	Do not Reject H0	2.173052
u1000_01	12.5	0.000018	Reject H0	2.794577	106.5	0.408235	Do not Reject H0	0.349778	209.5	0.999977	Do not Reject H0	2.597393
u1000_02	23.5	0.000096	Reject H0	1.640641	69	0.033469	Do not Reject H0	0.084709	192	0.999576	Do not Reject H0	1.696990
u1000_03	30	0.000324	Reject H0	1.473053	89	0.167902	Do not Reject H0	0.094474	186	0.998978	Do not Reject H0	1.490068
u1000_04	33.5	0.000524	Reject H0	1.541328	64	0.022081	Do not Reject H0	0.574183	178.5	0.997125	Do not Reject H0	1.388634
u1000_05	16	0.000030	Reject H0	1.861949	59	0.013272	Reject H0	0.851431	189.5	0.999369	Do not Reject H0	1.638601
u1000_06	27.5	0.000192	Reject H0	1.533356	61	0.016166	Reject H0	0.718530	176.5	0.996310	Do not Reject H0	1.322535
u1000_07	12	0.000008	Reject H0	1.901486	14	0.000013	Reject H0	1.791450	174	0.995051	Do not Reject H0	1.451556
u1000_08	14	0.000020	Reject H0	1.974853	52	0.005654	Reject H0	0.692564	186.5	0.999044	Do not Reject H0	1.730348
u1000_09	20.5	0.000058	Reject H0	1.595195	31	0.000320	Reject H0	1.036164	161	0.979389	Do not Reject H0	1.287768
u1000_10	20.5	0.000065	Reject H0	1.871652	70.5	0.040216	Do not Reject H0	0.549523	196	0.999762	Do not Reject H0	1.725597
u1000_11	17	0.000037	Reject H0	1.939606	65	0.024114	Do not Reject H0	0.058107	200.5	0.999883	Do not Reject H0	2.068546
u1000_12	18	0.000043	Reject H0	1.682263	58	0.011847	Reject H0	0.926263	182	0.998193	Do not Reject H0	1.472959
u1000_13	19	0.000050	Reject H0	2.025497	31	0.000347	Reject H0	0.767622	186.5	0.999025	Do not Reject H0	1.525024
u1000_14	8	0.000008	Reject H0	1.939013	52.5	0.006348	Reject H0	0.818435	203	0.999924	Do not Reject H0	1.771223
u1000_15	5	0.000004	Reject H0	2.884246	53	0.006918	Reject H0	0.752130	207.5	0.999965	Do not Reject H0	2.201898
u1000_16	11.5	0.000013	Reject H0	3.049221	66.5	0.027569	Do not Reject H0	0.702493	205	0.999945	Do not Reject H0	2.821565
u1000_17	14.5	0.000022	Reject H0	1.874430	70.5	0.040528	Do not Reject H0	0.369264	201.5	0.999903	Do not Reject H0	1.791240
u1000_18	14	0.000020	Reject H0	2.285342	37.5	0.000912	Reject H0	1.006121	186.5	0.999010	Do not Reject H0	1.639426
u1000_19	5.5	0.000004	Reject H0	2.367570	43	0.001701	Reject H0	0.190402	210	0.999979	Do not Reject H0	2.592763
HARD0	72.5	0.043051	Do not Reject H0	0.629156	71	0.034052	Do not Reject H0	0.682415	108.5	0.439230	Do not Reject H0	0.000000
HARD1	6	0.000002	Reject H0	2.935449	64	0.013942	Reject H0	0.796463	197.5	0.999879	Do not Reject H0	1.642213
HARD2	27.5	0.000109	Reject H0	1.572889	61.5	0.010662	Reject H0	0.805551	178.5	0.997907	Do not Reject H0	1.090175
HARD3	10	0.000004	Reject H0	2.466900	49	0.002510	Reject H0	1.090603	190	0.999833	Do not Reject H0	1.409657
HARD4	18	0.000014	Reject H0	2.054814	78.5	0.047914	Do not Reject H0	0.656469	180	0.998110	Do not Reject H0	1.151509
HARD5	3.5	0.000001	Reject H0	2.710530	44.5	0.000863	Reject H0	1.234846	204	0.999968	Do not Reject H0	1.648189
HARD6	46.5	0.001746	Reject H0	1.129458	54.5	0.004599	Reject H0	1.031127	125.5	0.721286	Do not Reject H0	0.305486
HARD7	20	0.000028	Reject H0	1.880515	61.5	0.008909	Reject H0	0.890473	182	0.998781	Do not Reject H0	1.184028
HARD8	21.5	0.000040	Reject H0	1.880515	73	0.035141	Do not Reject H0	0.683421	179	0.997907	Do not Reject H0	1.145192
HARD9	25	0.000040	Reject H0	1.513281	64.5	0.011796	Reject H0	0.429414	155	0.972881	Do not Reject H0	0.725200

Table A.19: GC-1BPP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
u120_00	25	0.0000887	Reject H0	1.6752380	150	0.9925324	Do not Reject H0	0.6687850	44	0.0014574	Reject H0	0.9359661
u120_01	4.5	0.0000025	Reject H0	3.3567473	91.5	0.1760707	Do not Reject H0	0.2138057	41	0.0011295	Reject H0	1.1733629
u120_02	10.5	0.0000042	Reject H0	2.4162581	106	0.3503752	Do not Reject H0	0.0877901	33.5	0.0001743	Reject H0	1.5501257
u120_03	0	0.0000003	Reject H0	-9.5708272	90	0.0398626	Do not Reject H0	0.6260334	30	0.0000398	Reject H0	1.4291798
u120_04	0	0.0000008	Reject H0	-5.0521065	78	0.0624145	Do not Reject H0	0.5270487	40.5	0.0010177	Reject H0	1.2160583
u120_05	10.5	0.0000048	Reject H0	2.6664303	74	0.0299831	Do not Reject H0	0.4847899	61	0.0084058	Reject H0	0.6396663
u120_06	19	0.0000206	Reject H0	2.2462483	75.5	0.0379476	Do not Reject H0	0.5388630	28.5	0.0001140	Reject H0	1.3817182
u120_07	2.5	0.0000012	Reject H0	2.7607364	102.5	0.3052143	Do not Reject H0	0.1444904	29.5	0.0001587	Reject H0	1.2290617
u120_08	5.5	0.0000031	Reject H0	2.4799939	125	0.7565821	Do not Reject H0	0.2071265	29.5	0.0001900	Reject H0	1.6250356
u120_09	22	0.0000380	Reject H0	2.0543069	100	0.2608660	Do not Reject H0	0.0000000	57.5	0.0065038	Reject H0	0.5011171
u120_10	6.5	0.0000033	Reject H0	2.2893309	64	0.0114074	Reject H0	0.6421294	31	0.0002048	Reject H0	1.3690245
u120_11	17.5	0.0000183	Reject H0	2.5364218	99	0.2721488	Do not Reject H0	0.3711156	22	0.0000595	Reject H0	1.6884019
u120_12	6.5	0.0000032	Reject H0	3.0100116	123	0.6971090	Do not Reject H0	0.2708659	43.5	0.0016726	Reject H0	1.0208474
u120_13	8	0.0000016	Reject H0	3.4818035	69	0.0107240	Reject H0	0.6907423	29	0.0000626	Reject H0	1.3255954
u120_14	12	0.0000054	Reject H0	2.7941788	90	0.1494297	Do not Reject H0	0.5445246	12	0.0000051	Reject H0	2.5592138
u120_15	16	0.0000061	Reject H0	2.9643463	99	0.1807073	Do not Reject H0	0.1671963	37.5	0.0002353	Reject H0	1.1540929
u120_16	1	0.0000006	Reject H0	4.3802411	137.5	0.8867457	Do not Reject H0	0.2621482	42.5	0.0012324	Reject H0	1.1200058
u120_17	25	0.0000642	Reject H0	1.8866689	105	0.3588476	Do not Reject H0	0.1057442	32	0.0002310	Reject H0	1.1649701
u120_18	13	0.0000096	Reject H0	2.5609520	106.5	0.3890146	Do not Reject H0	0.0951956	49.5	0.0029470	Reject H0	0.9937893
u120_19	0	0.0000006	Reject H0	4.3001018	75.5	0.0379476	Do not Reject H0	0.5573921	24	0.0000551	Reject H0	1.6358412
u250_00	4	0.0000027	Reject H0	2.7897573	97	0.2504273	Do not Reject H0	0.2068948	27	0.0001435	Reject H0	1.2295051
u250_01	47	0.0030367	Reject H0	1.4746905	140.5	0.8941827	Do not Reject H0	0.4799335	31.5	0.0002647	Reject H0	1.2587049
u250_02	8	0.0000032	Reject H0	3.6088675	71	0.0211687	Do not Reject H0	0.5247429	24	0.0000568	Reject H0	1.6335472
u250_03	4	0.0000027	Reject H0	3.1025493	70.5	0.0330913	Do not Reject H0	0.5162915	62.5	0.0159208	Reject H0	0.6175320
u250_04	13.5	0.0000170	Reject H0	2.7734926	67	0.0255973	Do not Reject H0	0.3959540	28	0.0002000	Reject H0	1.2997865
u250_05	12	0.0000058	Reject H0	2.1918806	50.5	0.0019464	Reject H0	0.9939615	28.5	0.0000841	Reject H0	1.5105034
u250_06	0.5	0.0000005	Reject H0	2.7826697	58.5	0.0029503	Reject H0	0.9218473	17.5	0.0000100	Reject H0	1.7955073
u250_07	12	0.0000080	Reject H0	3.2189226	78.5	0.0626395	Do not Reject H0	0.1570437	37	0.0006561	Reject H0	0.9940839
u250_08	2.5	0.0000015	Reject H0	3.3649801	79.5	0.0636210	Do not Reject H0	0.3663086	59.5	0.0089256	Reject H0	0.6389591
u250_09	1.5	0.0000017	Reject H0	2.9081155	64	0.0162814	Reject H0	0.7166836	30.5	0.0002311	Reject H0	1.1611174
u250_10	14	0.0000196	Reject H0	2.0410486	120.5	0.6475800	Do not Reject H0	0.3103396	32.5	0.0004204	Reject H0	1.2842588
u250_11	13	0.0000147	Reject H0	2.6003002	118	0.6111184	Do not Reject H0	0.0972353	48	0.0031479	Reject H0	0.9701432
u250_12	17	0.0000217	Reject H0	2.2806063	94.5	0.1999097	Do not Reject H0	0.3294656	52	0.0036125	Reject H0	1.0408188
u250_13	15.5	0.0000193	Reject H0	2.7173954	59	0.0067451	Reject H0	0.7956988	28	0.0000880	Reject H0	1.3025530
u250_14	9.5	0.0000051	Reject H0	2.8708643	72.5	0.0311097	Do not Reject H0	0.7906512	8	0.0000038	Reject H0	1.8356964
u250_15	14	0.0000127	Reject H0	2.6025948	116	0.5739672	Do not Reject H0	0.1783091	45.5	0.0024371	Reject H0	0.9255110
u250_16	10	0.0000062	Reject H0	2.7973476	96	0.2137720	Do not Reject H0	0.4331962	57.5	0.0074587	Reject H0	0.8619381
u250_17	10.5	0.0000065	Reject H0	2.4997248	81	0.0670132	Do not Reject H0	0.5135767	15.5	0.0000144	Reject H0	1.3729304
u250_18	0	0.0000006	Reject H0	3.3546180	86	0.0909656	Do not Reject H0	0.5249805	22	0.0000290	Reject H0	1.0903751
u250_19	1	0.0000009	Reject H0	3.3558279	76.5	0.0438471	Do not Reject H0	0.4762159	29.5	0.0001490	Reject H0	1.3725630

Table A.20: GC-1BPP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
u500_00	2.5	0.000002	Reject H0	2.521001	57.5	0.008511	Reject H0	0.585789	42.5	0.001722	Reject H0	1.179432
u500_01	5.5	0.000004	Reject H0	3.782786	47.5	0.002423	Reject H0	0.954266	24	0.000093	Reject H0	1.580211
u500_02	0.5	0.000001	Reject H0	3.003179	38	0.000539	Reject H0	0.783512	14.5	0.000017	Reject H0	1.196198
u500_03	11	0.000010	Reject H0	2.514444	105	0.380276	Do not Reject H0	0.326171	33	0.000440	Reject H0	1.250234
u500_04	0	0.000001	Reject H0	3.569843	48	0.002560	Reject H0	0.718623	16.5	0.000026	Reject H0	1.725534
u500_05	3.5	0.000003	Reject H0	3.773887	95.5	0.242250	Do not Reject H0	0.154676	75.5	0.061365	Do not Reject H0	0.435460
u500_06	11.5	0.000012	Reject H0	2.240724	80	0.078690	Do not Reject H0	0.344713	10	0.000009	Reject H0	1.405495
u500_07	5	0.000003	Reject H0	3.430363	53.5	0.005208	Reject H0	0.832850	30.5	0.000268	Reject H0	1.311110
u500_08	6.5	0.000005	Reject H0	2.157790	63	0.017390	Do not Reject H0	0.539156	14.5	0.000021	Reject H0	1.506167
u500_09	0.5	0.000002	Reject H0	2.389559	84.5	0.119775	Do not Reject H0	0.539714	42	0.001580	Reject H0	0.985290
u500_10	13	0.000016	Reject H0	2.138391	86	0.134987	Do not Reject H0	0.263950	25.5	0.000149	Reject H0	1.403637
u500_11	0	0.000001	Reject H0	2.826154	57	0.007160	Reject H0	0.911278	34	0.000370	Reject H0	1.040162
u500_12	4.5	0.000003	Reject H0	4.176322	48.5	0.002713	Reject H0	0.818680	17.5	0.000032	Reject H0	1.313395
u500_13	9.5	0.000009	Reject H0	1.898174	109	0.448792	Do not Reject H0	0.025978	48	0.003574	Reject H0	0.953955
u500_14	9	0.000008	Reject H0	2.432021	74.5	0.055696	Do not Reject H0	0.282922	42.5	0.001748	Reject H0	0.747961
u500_15	12.5	0.000014	Reject H0	2.955023	64.5	0.021788	Do not Reject H0	0.617884	50	0.004394	Reject H0	0.798422
u500_16	25	0.000130	Reject H0	2.020812	115.5	0.559641	Do not Reject H0	0.086536	42	0.001654	Reject H0	0.669489
u500_17	0	0.000001	Reject H0	2.839260	71	0.032432	Do not Reject H0	0.633956	2.5	0.000002	Reject H0	1.279322
u500_18	6.5	0.000004	Reject H0	2.830841	76.5	0.061061	Do not Reject H0	0.623528	8	0.000006	Reject H0	1.613405
u500_19	0	0.000001	Reject H0	3.718234	65.5	0.022604	Do not Reject H0	0.594748	35.5	0.000673	Reject H0	1.136165
u1000_00	7.5	0.000007	Reject H0	2.412417	69	0.034736	Do not Reject H0	0.499613	58.5	0.012394	Reject H0	0.783416
u1000_01	5	0.000004	Reject H0	2.986602	52.5	0.005828	Reject H0	0.665572	42	0.001412	Reject H0	0.870432
u1000_02	10.5	0.000011	Reject H0	1.929505	71.5	0.040043	Do not Reject H0	0.488104	34	0.000505	Reject H0	1.291841
u1000_03	3	0.000002	Reject H0	2.015554	33.5	0.000403	Reject H0	1.142390	7.5	0.000005	Reject H0	1.880089
u1000_04	26.5	0.000179	Reject H0	1.637511	84.5	0.119441	Do not Reject H0	0.383109	44.5	0.002376	Reject H0	1.029604
u1000_05	4	0.000003	Reject H0	2.002100	68.5	0.032318	Do not Reject H0	0.756059	22	0.000087	Reject H0	1.594399
u1000_06	16.5	0.000034	Reject H0	1.679109	55.5	0.007850	Reject H0	0.741187	27.5	0.000209	Reject H0	1.371482
u1000_07	7	0.000006	Reject H0	1.990725	65	0.015358	Reject H0	0.641031	4	0.000003	Reject H0	2.138047
u1000_08	4	0.000003	Reject H0	2.091543	46	0.002530	Reject H0	0.696743	19.5	0.000056	Reject H0	1.109112
u1000_09	7.5	0.000006	Reject H0	1.712965	49.5	0.003795	Reject H0	0.473547	12.5	0.000016	Reject H0	1.537669
u1000_10	9	0.000009	Reject H0	2.076972	59.5	0.012863	Reject H0	0.730275	31	0.000344	Reject H0	1.464735
u1000_11	7	0.000005	Reject H0	2.309775	82	0.090652	Do not Reject H0	0.460759	34	0.000460	Reject H0	1.404760
u1000_12	7	0.000006	Reject H0	1.778531	63	0.018278	Do not Reject H0	0.792471	31.5	0.000387	Reject H0	1.397553
u1000_13	11.5	0.000013	Reject H0	2.248563	83	0.102229	Do not Reject H0	0.448446	15	0.000023	Reject H0	1.193448
u1000_14	0.5	0.000002	Reject H0	2.111717	69	0.033485	Do not Reject H0	0.792270	16	0.000030	Reject H0	2.033644
u1000_15	0	0.000001	Reject H0	3.018290	91	0.182705	Do not Reject H0	0.544543	29.5	0.000255	Reject H0	0.968741
u1000_16	3.5	0.000003	Reject H0	3.273002	60.5	0.011992	Reject H0	0.778123	25.5	0.000134	Reject H0	1.530162
u1000_17	0.5	0.000001	Reject H0	2.095790	44	0.001395	Reject H0	0.814370	16	0.000027	Reject H0	1.607774
u1000_18	3.5	0.000003	Reject H0	2.541887	64.5	0.017474	Do not Reject H0	0.705830	11	0.000010	Reject H0	1.524401
u1000_19	2	0.000002	Reject H0	3.091136	84	0.111616	Do not Reject H0	0.432446	19.5	0.000050	Reject H0	1.428616
HARD0	26	0.000037	Reject H0	1.683081	79	0.038799	Do not Reject H0	0.789715	25	0.000027	Reject H0	2.000563
HARD1	1	0.000001	Reject H0	2.931067	85.5	0.078215	Do not Reject H0	0.561863	43.5	0.000903	Reject H0	1.157115
HARD2	35.5	0.000336	Reject H0	1.415600	132	0.847453	Do not Reject H0	0.409546	78	0.056901	Do not Reject H0	0.497391
HARD3	15	0.000012	Reject H0	2.036960	130.5	0.803124	Do not Reject H0	0.269997	66.5	0.023434	Do not Reject H0	0.742491
HARD4	20	0.000015	Reject H0	1.981839	106	0.316293	Do not Reject H0	0.000000	75.5	0.030670	Do not Reject H0	0.626838
HARD5	10.5	0.000006	Reject H0	2.231012	122	0.712492	Do not Reject H0	0.248696	62.5	0.013605	Reject H0	0.789715
HARD6	53	0.004278	Reject H0	0.960362	115	0.558999	Do not Reject H0	0.152294	61.5	0.011110	Reject H0	0.817361
HARD7	29	0.000161	Reject H0	1.587970	128.5	0.794723	Do not Reject H0	0.351160	79	0.065977	Do not Reject H0	0.526741
HARD8	20.5	0.000022	Reject H0	1.900505	100	0.237896	Do not Reject H0	0.112637	66	0.013251	Reject H0	0.739946
HARD9	30	0.000188	Reject H0	1.335453	95	0.205621	Do not Reject H0	0.400593	63	0.016066	Reject H0	0.609177

Table A.21: GC-MSSP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
C_S_0_I0	14.68888889	0.002102772925	Reject H0
C_S_0_I1	8.634146341	0.03457213776	Reject H0
C_S_0_I2	20.875	0.0001117599264	Reject H0
C_S_0_I3	11.65957447	0.008645123995	Reject H0
C_S_0_I4	17.59285714	0.0005336147049	Reject H0
C_S_1_I0	9.653061224	0.02175757746	Reject H0
C_S_1_I1	12.84	0.004995706044	Reject H0
C_S_1_I2	14.08029197	0.002797873977	Reject H0
C_S_1_I3	18.18243243	0.0004033388184	Reject H0
C_S_1_I4	13.6122449	0.00348340853	Reject H0
C_S_2_I0	20.05714286	0.0001651761692	Reject H0
C_S_2_I1	17.23448276	0.0006324413461	Reject H0
C_S_2_I2	5.612903226	0.1320395572	Do not Reject H0
C_S_2_I3	25.53061224	1.20E-05	Reject H0
C_S_2_I4	27.12244898	5.55E-06	Reject H0
C_S_3_I0	18.26174497	0.0003884211849	Reject H0
C_S_3_I1	11.16326531	0.01087501632	Reject H0
C_S_3_I2	11.96	0.007521447233	Reject H0
C_S_3_I3	5.16	0.160449136	Do not Reject H0
C_S_3_I4	14.35135135	0.002463895888	Reject H0

Table A.22: GC-MSSP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	18	0.00002	Reject H0	1.60906	94	0.21414	Do not Reject H0	0.43267	181.5	0.99931	Do not Reject H0	1.13385
C_S_0_I1	51	0.00291	Reject H0	0.66430	105	0.38180	Do not Reject H0	0.12898	165	0.99376	Do not Reject H0	0.69207
C_S_0_I2	15.5	0.00003	Reject H0	1.83932	93	0.21377	Do not Reject H0	0.35865	204.5	0.99994	Do not Reject H0	1.78873
C_S_0_I3	34	0.00054	Reject H0	1.82636	102	0.33217	Do not Reject H0	0.25946	189	0.99935	Do not Reject H0	1.84675
C_S_0_I4	13.5	0.00002	Reject H0	1.96390	106.5	0.40657	Do not Reject H0	0.15708	204.5	0.99995	Do not Reject H0	1.76811
C_S_1_I0	44.5	0.00254	Reject H0	1.00608	100	0.30906	Do not Reject H0	0.02464	178	0.99692	Do not Reject H0	0.93489
C_S_1_I1	46	0.00308	Reject H0	1.29114	90.5	0.18548	Do not Reject H0	0.28052	170.5	0.99245	Do not Reject H0	1.09024
C_S_1_I2	30	0.00026	Reject H0	1.37727	52	0.00411	Reject H0	0.94209	174	0.99618	Do not Reject H0	0.91922
C_S_1_I3	17	0.00004	Reject H0	2.08398	67.5	0.03230	Do not Reject H0	0.68245	198.5	0.99983	Do not Reject H0	1.60881
C_S_1_I4	35.5	0.00072	Reject H0	1.34232	114	0.53353	Do not Reject H0	0.17475	200	0.99988	Do not Reject H0	1.49426
C_S_2_I0	8.5	0.00001	Reject H0	1.50555	56.5	0.00912	Reject H0	0.87310	196	0.99977	Do not Reject H0	1.16085
C_S_2_I1	18	0.00004	Reject H0	1.37656	61	0.01362	Reject H0	0.91185	181.5	0.99821	Do not Reject H0	1.11617
C_S_2_I3	6	0.00000	Reject H0	1.61815	48.5	0.00417	Reject H0	0.84602	196	0.99979	Do not Reject H0	1.11871
C_S_2_I4	4	0.00000	Reject H0	2.54465	56	0.00971	Reject H0	0.82258	212.5	0.99999	Do not Reject H0	2.01494
C_S_3_I0	27	0.00021	Reject H0	1.64912	42	0.00184	Reject H0	1.16558	169.5	0.99148	Do not Reject H0	0.93942
C_S_3_I1	35	0.00065	Reject H0	1.14405	86.5	0.14054	Do not Reject H0	0.41110	174	0.99517	Do not Reject H0	0.94937
C_S_3_I2	39	0.00123	Reject H0	1.29027	62	0.01904	Do not Reject H0	0.79768	148	0.93231	Do not Reject H0	0.71277
C_S_3_I4	34.5	0.00065	Reject H0	1.45786	91	0.19181	Do not Reject H0	0.15766	172.5	0.99396	Do not Reject H0	1.07573

Table A.23: GC-MSSP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	42.5	0.00070	Reject H0	1.09741	129.5	0.77506	Do not Reject H0	0.32596	107	0.41521	Do not Reject H0	0.06324
C_S_0_I1	62	0.00930	Reject H0	0.62272	120	0.63377	Do not Reject H0	0.00113	117	0.58606	Do not Reject H0	0.11983
C_S_0_I2	29	0.00028	Reject H0	1.43372	137	0.85180	Do not Reject H0	0.32317	119	0.61475	Do not Reject H0	0.02716
C_S_0_I3	36.5	0.00077	Reject H0	1.58738	133.5	0.82638	Do not Reject H0	0.28469	125.5	0.72062	Do not Reject H0	0.11099
C_S_0_I4	19.5	0.00005	Reject H0	1.75105	142.5	0.90937	Do not Reject H0	0.55493	127	0.74298	Do not Reject H0	0.27233
C_S_1_I0	50	0.00504	Reject H0	0.90291	130	0.77260	Do not Reject H0	0.12560	118	0.59842	Do not Reject H0	0.08258
C_S_1_I1	69.5	0.03872	Do not Reject H0	0.89898	168	0.99008	Do not Reject H0	0.92759	135.5	0.83752	Do not Reject H0	0.47269
C_S_1_I2	65.5	0.02557	Do not Reject H0	0.71593	156.5	0.97349	Do not Reject H0	0.88512	114	0.53438	Do not Reject H0	0.15998
C_S_1_I3	29	0.00028	Reject H0	1.58620	144.5	0.91166	Do not Reject H0	0.47306	109	0.45028	Do not Reject H0	0.12156
C_S_1_I4	35.5	0.00072	Reject H0	1.23429	123	0.67702	Do not Reject H0	0.15119	122.5	0.67081	Do not Reject H0	0.33387
C_S_2_I0	28.5	0.00024	Reject H0	1.13975	157	0.97223	Do not Reject H0	0.71646	103.5	0.35772	Do not Reject H0	0.00873
C_S_2_I1	54.5	0.00795	Reject H0	1.07644	187.5	0.99929	Do not Reject H0	1.18417	136.5	0.85215	Do not Reject H0	0.15261
C_S_2_I3	15.5	0.00003	Reject H0	1.28764	158	0.97190	Do not Reject H0	0.56448	84	0.12245	Do not Reject H0	0.30765
C_S_2_I4	14	0.00002	Reject H0	1.92571	179	0.99750	Do not Reject H0	1.00012	119.5	0.62518	Do not Reject H0	0.17676
C_S_3_I0	37	0.00093	Reject H0	1.33264	138.5	0.86421	Do not Reject H0	0.46588	67	0.03097	Do not Reject H0	0.65290
C_S_3_I1	51	0.00554	Reject H0	1.02526	127.5	0.74272	Do not Reject H0	0.07001	107	0.41675	Do not Reject H0	0.29407
C_S_3_I2	53	0.00720	Reject H0	1.05552	139	0.86862	Do not Reject H0	0.39443	81.5	0.10290	Do not Reject H0	0.45769
C_S_3_I4	39.5	0.00130	Reject H0	1.40977	122.5	0.66858	Do not Reject H0	0.06144	91.5	0.19732	Do not Reject H0	0.11192

Table A.24: GP-CVRP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
A-n32-k5	11.53691275	0.009150185697	Reject H0
A-n33-k5	7.979591837	0.04643536921	Reject H0
A-n33-k6	5.093959732	0.1650448337	Do not Reject H0
A-n34-k5	11.61486486	0.008825946623	Reject H0
A-n36-k5	7.714285714	0.05230077986	Do not Reject H0
A-n37-k5	19.06711409	0.0002648004892	Reject H0
A-n37-k6	0.6891891892	0.8757437673	Do not Reject H0
A-n38-k5	15.14189189	0.001699320833	Reject H0
A-n39-k5	9.604026846	0.02225005654	Reject H0
A-n39-k6	14.83892617	0.00195960171	Reject H0
A-n44-k6	16.84	0.0007623585042	Reject H0
A-n45-k6	15.40268456	0.001502944888	Reject H0
A-n45-k7	11.24	0.01049643382	Reject H0
A-n46-k7	3.32	0.3448687036	Do not Reject H0
A-n48-k7	5.32	0.1498098733	Do not Reject H0
A-n53-k7	8.12	0.04359568836	Reject H0
A-n54-k7	16.93288591	0.0007295626648	Reject H0
A-n55-k9	15.76510067	0.001266912498	Reject H0
A-n60-k9	6.2	0.1022750265	Do not Reject H0
A-n61-k9	9.608108108	0.02220865215	Reject H0
A-n62-k8	30.84	9.19E-07	Reject H0
A-n63-k9	0.2214765101	0.9740499804	Do not Reject H0
A-n63-k10	3.222972973	0.3585080532	Do not Reject H0
A-n64-k9	3.926174497	0.2695470909	Do not Reject H0
A-n65-k9	12.48648649	0.005889572533	Reject H0
A-n69-k9	20.52	0.0001324235234	Reject H0
A-n80-k10	1.8	0.6149349358	Do not Reject H0
M-n101-k10	10.36	0.01574130184	Reject H0
M-n121-k7	29.48	1.78E-06	Reject H0
M-n151-k12	15.28187919	0.001590941152	Reject H0
M-n200-k16	17.32	0.0006073210476	Reject H0
M-n200-k17	17.2147651	0.0006383782806	Reject H0

Table A.25: GP-CVRP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
A-n32-k5	184	0.99859	Do not Reject H0	0.86485	97	0.26685	Do not Reject H0	0.10339	38.5	0.00115	Reject H0	0.92580
A-n33-k5	176.5	0.99640	Do not Reject H0	1.25783	122	0.66314	Do not Reject H0	0.14293	52.5	0.00665	Reject H0	1.16818
A-n34-k5	191.5	0.99952	Do not Reject H0	1.41443	92	0.20311	Do not Reject H0	0.31487	27	0.00021	Reject H0	1.61762
A-n37-k5	187	0.99909	Do not Reject H0	1.30141	127.5	0.74044	Do not Reject H0	0.16765	38	0.00104	Reject H0	1.23381
A-n38-k5	202	0.99991	Do not Reject H0	1.82251	104	0.36977	Do not Reject H0	0.22082	17.5	0.00004	Reject H0	2.20126
A-n39-k5	183.5	0.99849	Do not Reject H0	1.31673	103.5	0.36205	Do not Reject H0	0.01733	44	0.00239	Reject H0	1.36051
A-n39-k6	192	0.99955	Do not Reject H0	1.29710	83	0.11436	Do not Reject H0	0.39877	31	0.00039	Reject H0	1.46958
A-n44-k6	194.5	0.99969	Do not Reject H0	1.65695	112.5	0.50827	Do not Reject H0	0.01425	26.5	0.00019	Reject H0	1.69630
A-n45-k6	179.5	0.99745	Do not Reject H0	1.10421	79.5	0.08851	Do not Reject H0	0.42956	30	0.00033	Reject H0	1.33611
A-n45-k7	173.5	0.99465	Do not Reject H0	0.81010	91	0.19173	Do not Reject H0	0.28089	48	0.00396	Reject H0	0.92819
A-n53-k7	181	0.99790	Do not Reject H0	0.83600	96.5	0.25998	Do not Reject H0	0.01923	49	0.00447	Reject H0	0.76395
A-n54-k7	188.5	0.99925	Do not Reject H0	1.35159	107	0.41782	Do not Reject H0	0.17271	29	0.00029	Reject H0	1.48795
A-n55-k9	194	0.99967	Do not Reject H0	1.26485	105.5	0.39365	Do not Reject H0	0.05323	35.5	0.00075	Reject H0	1.20954
A-n61-k9	157	0.96912	Do not Reject H0	0.60586	76	0.06736	Do not Reject H0	0.32108	58	0.01250	Reject H0	0.70990
A-n62-k8	225	1.00000	Do not Reject H0	2.98833	83.5	0.11840	Do not Reject H0	0.50040	0	0.00000	Reject H0	3.35084
A-n65-k9	178.5	0.99711	Do not Reject H0	1.32966	92	0.20321	Do not Reject H0	0.10925	41.5	0.00172	Reject H0	1.31712
A-n69-k9	206	0.99995	Do not Reject H0	1.90986	105	0.38569	Do not Reject H0	0.12150	17	0.00004	Reject H0	1.98985
M-n101-k10	176.5	0.99628	Do not Reject H0	1.02158	151	0.94744	Do not Reject H0	0.52521	51	0.00569	Reject H0	0.82511
M-n121-k7	225	1.00000	Do not Reject H0	3.12680	82	0.10664	Do not Reject H0	0.59918	0	0.00000	Reject H0	4.22138
M-n151-k12	193	0.99961	Do not Reject H0	1.49731	105	0.38566	Do not Reject H0	0.30972	27	0.00021	Reject H0	1.67371
M-n200-k16	195	0.99971	Do not Reject H0	1.43216	124	0.69076	Do not Reject H0	0.02741	26	0.00018	Reject H0	1.47875
M-n200-k17	192.5	0.99958	Do not Reject H0	1.50743	63.5	0.02209	Do not Reject H0	0.84424	14.5	0.00003	Reject H0	1.81949

Table A.26: GP-CVRP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
A-n32-k5	180	0.99761	Do not Reject H0	0.84853	122.5	0.66848	Do not Reject H0	0.04584	110.5	0.47518	Do not Reject H0	0.04224
A-n33-k5	167	0.98876	Do not Reject H0	0.91276	102.5	0.34598	Do not Reject H0	0.35660	108.5	0.44209	Do not Reject H0	0.25040
A-n34-k5	169.5	0.99151	Do not Reject H0	0.85011	84.5	0.12659	Do not Reject H0	0.49868	69	0.03710	Do not Reject H0	0.73293
A-n37-k5	169	0.99114	Do not Reject H0	0.60287	44.5	0.00254	Reject H0	0.98845	49	0.00433	Reject H0	0.88843
A-n38-k5	175.5	0.99582	Do not Reject H0	1.01970	65.5	0.02666	Do not Reject H0	0.64891	56.5	0.01052	Reject H0	0.89405
A-n39-k5	178.5	0.99711	Do not Reject H0	1.24854	107	0.41767	Do not Reject H0	0.18438	95	0.24015	Do not Reject H0	0.23105
A-n39-k6	193	0.99961	Do not Reject H0	1.42773	144.5	0.91132	Do not Reject H0	0.33973	115.5	0.55774	Do not Reject H0	0.01759
A-n44-k6	190	0.99939	Do not Reject H0	1.58188	102	0.33908	Do not Reject H0	0.10910	97.5	0.27371	Do not Reject H0	0.12788
A-n45-k6	166	0.98746	Do not Reject H0	0.58385	73.5	0.05508	Do not Reject H0	0.64594	58	0.01251	Reject H0	0.97598
A-n45-k7	184.5	0.99869	Do not Reject H0	0.97096	149	0.93763	Do not Reject H0	0.35568	123.5	0.68341	Do not Reject H0	0.02599
A-n53-k7	161	0.97900	Do not Reject H0	0.52163	91	0.19163	Do not Reject H0	0.41144	87	0.14974	Do not Reject H0	0.33012
A-n54-k7	191	0.99947	Do not Reject H0	1.41164	126.5	0.72629	Do not Reject H0	0.13602	124	0.69070	Do not Reject H0	0.00715
A-n55-k9	180.5	0.99777	Do not Reject H0	0.91257	78.5	0.08213	Do not Reject H0	0.53839	84.5	0.12682	Do not Reject H0	0.47820
A-n61-k9	142	0.89344	Do not Reject H0	0.47206	85.5	0.13531	Do not Reject H0	0.37110	58	0.01245	Reject H0	0.56729
A-n62-k8	224	1.00000	Do not Reject H0	2.65221	103.5	0.36212	Do not Reject H0	0.14971	76	0.06763	Do not Reject H0	0.59152
A-n65-k9	184.5	0.99868	Do not Reject H0	1.35170	124.5	0.69800	Do not Reject H0	0.17913	112	0.50000	Do not Reject H0	0.05598
A-n69-k9	199	0.99985	Do not Reject H0	1.55134	102	0.33905	Do not Reject H0	0.30077	102	0.33907	Do not Reject H0	0.39374
M-n101-k10	164	0.98456	Do not Reject H0	0.63848	62	0.01888	Do not Reject H0	0.81180	90.5	0.18567	Do not Reject H0	0.38096
M-n121-k7	197.5	0.99981	Do not Reject H0	1.47086	74	0.05745	Do not Reject H0	0.70547	55.5	0.00954	Reject H0	1.09447
M-n151-k12	180	0.99760	Do not Reject H0	1.20880	71.5	0.04646	Do not Reject H0	0.54428	53	0.00715	Reject H0	0.89969
M-n200-k16	184	0.99859	Do not Reject H0	1.22825	72.5	0.05063	Do not Reject H0	0.47054	73	0.05275	Do not Reject H0	0.56594
M-n200-k17	177	0.99650	Do not Reject H0	1.15346	76.5	0.07040	Do not Reject H0	0.61739	31.5	0.00042	Reject H0	1.30395

Table A.27: GP-MSSP Friedman Test Results

	Friedman Test		
Instance	χ^2	P Value	Outcome
C_S_0_I0	19.64	0.0002015427626	Reject H0
C_S_0_I1	20.28	0.000148506523	Reject H0
C_S_0_I2	21.21582734	9.50E-05	Reject H0
C_S_0_I3	17.18181818	0.0006484223631	Reject H0
C_S_0_I4	15.56	0.001395557823	Reject H0
C_S_1_I0	21.32	9.03E-05	Reject H0
C_S_1_I1	8.154362416	0.04292693149	Reject H0
C_S_1_I2	20.68	0.0001226773353	Reject H0
C_S_1_I3	18.36	0.0003706995932	Reject H0
C_S_1_I4	4.30952381	0.2299226788	Do not Reject H0
C_S_2_I0	9.805369128	0.02029512609	Reject H0
C_S_2_I1	17.93283582	0.0004541037107	Reject H0
C_S_2_I2	16.96240602	0.0007194363456	Reject H0
C_S_2_I3	17.23943662	0.0006309583976	Reject H0
C_S_2_I4	8.04	0.0451922194	Reject H0
C_S_3_I0	10.45652174	0.01505875444	Reject H0
C_S_3_I1	17.93233083	0.0004542126054	Reject H0
C_S_3_I2	8.355704698	0.03920487491	Reject H0
C_S_3_I3	16.97916667	0.0007137492093	Reject H0
C_S_3_I4	8.310810811	0.04000649308	Reject H0

Table A.28: GP-MSSP Mann-Whitney U Tests with 1D, 2D and 3D Pheromone Maps

	Mann-Whitney U Test: 2D–1D				Mann-Whitney U Test: 2D–3D				Mann-Whitney U Test: 1D–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	19	0.00006	Reject H0	1.79936	84	0.12264	Do not Reject H0	0.24132	202	0.99991	Do not Reject H0	1.63778
C_S_0_I1	16	0.00003	Reject H0	1.88513	160	0.97677	Do not Reject H0	0.66496	213	0.99999	Do not Reject H0	2.15988
C_S_0_I2	26	0.00016	Reject H0	1.21237	149	0.94241	Do not Reject H0	0.53089	210.5	0.99998	Do not Reject H0	1.64137
C_S_0_I3	55.5	0.00316	Reject H0	0.92022	150	0.97044	Do not Reject H0	0.66947	189.5	0.99973	Do not Reject H0	1.06529
C_S_0_I4	34	0.00061	Reject H0	1.42530	71	0.04449	Do not Reject H0	0.64999	179	0.99727	Do not Reject H0	1.07115
C_S_1_I0	11.5	0.00002	Reject H0	1.79242	119.5	0.62216	Do not Reject H0	0.14250	213	0.99999	Do not Reject H0	1.82295
C_S_1_I1	72.5	0.05065	Do not Reject H0	0.30738	164.5	0.98529	Do not Reject H0	0.84967	168	0.98990	Do not Reject H0	0.73216
C_S_1_I2	20	0.00007	Reject H0	1.34031	101	0.32408	Do not Reject H0	0.04403	187	0.99907	Do not Reject H0	1.05577
C_S_1_I3	30	0.00034	Reject H0	1.22469	64	0.02324	Do not Reject H0	0.76690	183	0.99838	Do not Reject H0	0.75628
C_S_2_I0	64.5	0.02434	Do not Reject H0	0.78830	109.5	0.45864	Do not Reject H0	0.00184	165	0.98608	Do not Reject H0	0.83443
C_S_2_I1	74	0.05003	Do not Reject H0	0.58773	49	0.00317	Reject H0	0.90915	100.5	0.31278	Do not Reject H0	0.14501
C_S_2_I2	29.5	0.00021	Reject H0	1.50548	142.5	0.90751	Do not Reject H0	0.27031	201.5	0.99991	Do not Reject H0	1.64220
C_S_2_I3	59.5	0.01220	Reject H0	0.94213	132	0.80467	Do not Reject H0	0.31864	173.5	0.99534	Do not Reject H0	1.01704
C_S_2_I4	59	0.01393	Reject H0	0.84261	74.5	0.05990	Do not Reject H0	0.58781	144	0.90789	Do not Reject H0	0.39791
C_S_3_I0	62	0.01718	Do not Reject H0	0.95645	79	0.06854	Do not Reject H0	0.69753	147.5	0.93077	Do not Reject H0	0.49094
C_S_3_I1	53.5	0.00597	Reject H0	1.54852	112.5	0.50934	Do not Reject H0	0.19162	176	0.99639	Do not Reject H0	1.57507
C_S_3_I2	55.5	0.00954	Reject H0	0.93444	122	0.66087	Do not Reject H0	0.03086	172.5	0.99396	Do not Reject H0	0.92108
C_S_3_I3	135.5	0.84314	Do not Reject H0	0.41304	48	0.00304	Reject H0	0.77074	56.5	0.01041	Reject H0	0.73270
C_S_3_I4	99	0.29433	Do not Reject H0	0.10224	64.5	0.02222	Do not Reject H0	1.00951	101	0.32283	Do not Reject H0	0.34330

Table A.29: GP-MSSP Mann-Whitney U Tests with 1D, 2D, 3D and HACOH

	Mann-Whitney U Test: H-1D				Mann-Whitney U Test: H–2D				Mann-Whitney U Test: H–3D			
Instance	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size	U Value	P Value	Outcome	Effect Size
C_S_0_I0	27	0.00021	Reject H0	1.60617	135	0.83004	Do not Reject H0	0.45383	127	0.73334	Do not Reject H0	0.20305
C_S_0_I1	21	0.00008	Reject H0	1.89582	80.5	0.09558	Do not Reject H0	0.46581	119	0.61425	Do not Reject H0	0.13044
C_S_0_I2	25	0.00014	Reject H0	1.17435	113.5	0.52509	Do not Reject H0	0.00222	146	0.92602	Do not Reject H0	0.50872
C_S_0_I3	59.5	0.00451	Reject H0	0.89251	120	0.69630	Do not Reject H0	0.16310	157.5	0.98956	Do not Reject H0	0.85094
C_S_0_I4	35	0.00070	Reject H0	1.35223	113.5	0.52481	Do not Reject H0	0.02574	77.5	0.07619	Do not Reject H0	0.55990
C_S_1_I0	12	0.00002	Reject H0	1.87267	87	0.14977	Do not Reject H0	0.29915	100	0.30920	Do not Reject H0	0.15756
C_S_1_I1	91.5	0.19732	Do not Reject H0	0.12306	136.5	0.84551	Do not Reject H0	0.32068	181	0.99790	Do not Reject H0	1.01649
C_S_1_I2	16	0.00003	Reject H0	1.89586	79	0.08549	Do not Reject H0	0.20636	89.5	0.17529	Do not Reject H0	0.10180
C_S_1_I3	29	0.00029	Reject H0	1.14631	122	0.66085	Do not Reject H0	0.25829	70.5	0.04258	Do not Reject H0	0.63241
C_S_2_I0	114	0.53308	Do not Reject H0	0.10154	171.5	0.99326	Do not Reject H0	0.87647	171.5	0.99327	Do not Reject H0	0.93177
C_S_2_I1	50.5	0.00467	Reject H0	0.85645	73.5	0.04782	Do not Reject H0	0.51680	27.5	0.00017	Reject H0	1.26642
C_S_2_I2	39.5	0.00105	Reject H0	1.43058	137	0.87450	Do not Reject H0	0.14771	158	0.97585	Do not Reject H0	0.41508
C_S_2_I3	30	0.00032	Reject H0	1.16275	55.5	0.00893	Reject H0	0.80230	77.5	0.07523	Do not Reject H0	0.38646
C_S_2_I4	67	0.03093	Do not Reject H0	0.58896	138	0.85958	Do not Reject H0	0.16162	104.5	0.37785	Do not Reject H0	0.29599
C_S_3_I0	43	0.00203	Reject H0	1.21345	67.5	0.02422	Do not Reject H0	0.63883	46.5	0.00265	Reject H0	1.01597
C_S_3_I1	28	0.00022	Reject H0	1.85939	41	0.00057	Reject H0	0.97185	48	0.00281	Reject H0	0.79845
C_S_3_I2	61.5	0.01788	Do not Reject H0	0.74514	129.5	0.76656	Do not Reject H0	0.36719	140	0.87763	Do not Reject H0	0.36796
C_S_3_I3	85.5	0.13488	Do not Reject H0	0.37246	46.5	0.00249	Reject H0	0.91698	17	0.00004	Reject H0	1.25034
C_S_3_I4	82	0.10633	Do not Reject H0	0.44822	83.5	0.11792	Do not Reject H0	0.50909	26	0.00016	Reject H0	1.14962

Appendix B

Generated Heuristics

This appendix contains the results of the statistical tests that are consulted in Section 10.5.

B.1 HACO-GC

```
[-:{-:C,C}|{-:{-:C,{A:{-:C,{/:C,{-:S,{A:{-:F,C}}}}}}}|C}]
```

Fig. B.1: Heuristic Generated with 1D Pheromone Map

```
[/:{/:S,F}|{-:{-:{A:{A:{A:C}}}|{A:S}}|F}]
```

Fig. B.2: Heuristic Generated with 2D Pheromone Map

```
[*:{/:{/:{/:F,C}|{-:{*:F,S}|{*:{A:F}|S}}}|{-:C,{+:F,S}}}|C]
```

Fig. B.3: Heuristic Generated with 3D Pheromone Map

B.2 HACO-GP

```
[M12:S3,S9,{M6:{M12:{M9:S13,{M13:S10,S11}}|
S11,S5,S12,S5}|S9}|{M13:{M3:S8,S9}|{M11:S17,S4,S9,
{M2:S7,S13}|{M6:{M12:S11,S2,{M12:S9,S4,S3,S2,S9}|
{M11:S3,S8,{M13:{M12:{M11:S3,{M13:S5,
{M13:S8,S10}}|{M9:S9,S15}|S2,S7}|S2,{M10:S12,S9}|
{M13:{M3:{M13:S9,{M4:{M13:{M6:S4,{M8:S14,S9}}|S2}|
{M13:S12,S8}}}|{M13:S17,S8}}|S3}|{M12:S7,S5,
{M13:S2,S12}|S9,S6}}|{M13:S3,S8}}|S2,S10}|S3}|
{M13:S16,S2}}}}|S3]
```

Fig. B.4: Heuristic Generated with 1D Pheromone Map

```
[M6:{M11:{M4:{M13:{M1:{M13:{M4:S10,{M7:S1,{M8:S14,
{M1:{M7:S1,S3}|S12,S16}}}}|S2}|{M6:S14,S1}|{M2:S9,
{M7:S10,{M7:S8,{M5:{M2:S6,S11}|{M10:{M10:{M8:S6,
{M3:{M10:S2,S5}|{M4:{M13:S16,S7}|S1}}}|S17}|
S8}}}}}}|S12}|S4}|S9,S16,{M8:S3,S5}|{M2:S3,
{M12:S7,S4,S15,{M13:S10,S11}|{M6:S15,S8}}}}|S5]
```

Fig. B.5: Heuristic Generated with 2D Pheromone Map

```
[M12:{M9:{M8:{M7:S6,{M2:S9,S3}}|{M4:S6,S4}}|S4}|
{M13:{M9:S1,S3}|S10}|{M10:S10,{M8:{M3:{M10:S12,
{M9:S15,{M3:S13,S14}}}|{M6:S8,S17}}|{M2:
{M12:S5,S5,S8,{M11:{M7:S8,{M7:{M13:{M11:{M5:S2,
{M5:S10,S1}}|S6,S3,{M5:S4,S16}|{M4:{M6:S16,S14}|
{M3:S3,S8}}}|S15}|S16}}|S12,{M3:S2,S3}|S1,S3}|
{M10:S12,S2}}|S13}}}|S2,S3]
```

Fig. B.6: Heuristic Generated with 3D Pheromone Map