A Study of Ant-Based Pheromone Spaces for Generation Constructive Hyper-Heuristics

Emilio Singh

cnr Lynnwood Road and Roper Street, Pretoria

Nelishia Pillay

cnr Lynnwood Road and Roper Street, Pretoria

Abstract

Research into the applicability of ant-based optimisation techniques for hyper-heuristics is largely limited. This paper expands upon the existing body of research by presenting a novel ant-based generation constructive hyper-heuristic and then investigates how different pheromone maps affect its performance. Previous work has focused on applying ant-based optimisation techniques that work in the solution space directly to the heuristic space and we hypothesise that this may be problematic for the hyper-heuristic's efficacy. The focus of this analysis is primarily on how the pheromone map, 2D and 3D, of ant-based methods, can be used for this hyper-heuristic task. 2D pheromone maps are the predominant pheromone map type used by ant-based algorithms. Thus the comparison here is between the existing 2D pheromone map and the newly introduced 3D pheromone map. The analysis consists of multiple experiments with algorithms in the TSP and 1DBPP domain which are assessed in terms of optimality and generality. The results of the experiment demonstrate key differences in performance between the two different pheromone spaces. The 3D pheromone map showed better generality and optimality in the 1DBPP domain whereas the 2D pheromone map showed better generality and only marginally better optimality for the TSP domain. The analysis indicated that the different

[1]University of Pretoria,Corresponding Author u14006512@tuks.co.za
[2]University of Pretoria,npillay@cs.up.ac.za

pheromone maps work most optimally for different types of optimisation problems. The hybrid method showed some improvements in generality but showed little improvements in optimality overall.
Keywords: Generation constructive hyper-heuristics, Ant algorithms, Discrete combinatorial optimization

---

1. Introduction

Hyper-heuristics as a field concerns itself with providing more general solutions to combinatorial problems. It achieves this through operating in the heuristic space of problems rather than the solution space [1]. Solutions to problems are constructed by using heuristics which then solve the underlying problem. In particular, hyper-heuristics gave rise to generation constructive hyper-heuristics which is the use of a hyper-heuristic to generate new heuristics that can create solutions for a given problem [2]. In this way, the task of creating new heuristics, especially for problems that do not have them, is automated and may even give rise to heuristics that human creators would never consider.

Until now, generation constructive hyper-heuristics primarily make use of genetic programming (GP) as the primary search method [3] and when ant-based methods have been applied, they have been typically applied to selection hyper-heuristics [4]. In particular, the focus of the use of ant-based methods has been to use the ant-based technique on the heuristic space in the same way as the ant-based technique is used in the 2D solution space. The focus of this research is therefore to more thoroughly investigate the use of the pheromone space for hyper-heuristics, specifically generation constructive hyper-heuristics, and how these spaces affect the ability of a hyper-heuristic to perform.

This is achieved through a study of a native 2D pheromone map against a 3D projected pheromone map alongside an algorithm that hybridises the two spaces into a single hyper-heuristic as a culmination of the work. This study focuses on these two types of pheromone spaces because the 2D implementation is the common representation for ant-based methods with the 3D projection offering

a different representation that may provide benefits for the task at hand. The proposed ant-based hyper-heuristic is therefore called hyper-heuristic ant colony optimisation (HACO).

Therefore the paper has several major aims concerning its research. These are:

1. An investigation into using 2D pheromone maps in generation construction hyper-heuristics for discrete optimisation.
2. An investigation into using 3D pheromone maps in generation construction hyper-heuristics for discrete optimisation.
3. An investigation into using hybrid pheromone maps, using both 2D and 3D maps, in generation construction hyper-heuristics for discrete optimisation.

Principally, this research aims to examine the effect that different types of pheromone maps will have on the performance of an ant-based hyper-heuristic that uses them in its operation and to test the effect of the hybridisation of pheromone maps as well.

The rest of the paper is structured as follows. Section 2 provides a background to concepts pertinent to this research. In section 3 the use and expansion of pheromone spaces are explained. In section 4 the functioning of the HACO method is presented and explained in detail. The hybridisation of the two pheromone spaces is discussed in section 5. The experimental methodology used in this paper is presented in section 6. The results of the research with relevant discussion are presented in section 7. Finally, the research is concluded in section 8 along with some suggestions for future work.

2. Background

The purpose of this section is to provide background information on the core topics relevant to the research that is presented here.

2.1. Generation Constructive Hyper-Heuristics

Broadly speaking generation constructive hyper-heuristics concern themselves with finding ways to organise problem attributes into a heuristic capable of solving a given problem I or problems similar to I [2]. In terms of these induced heuristics, these can either be disposable (generated on the fly for a problem and discarded as needed) or reusable (generated and used on other problems). The attributes that make up the space searched by the hyper-heuristic should be as simple as possible [5]. These attributes form the core of the representation of any potential generated heuristic and should represent the essential characteristics of the underlying problem that is being solved. Often they will be derived from existing heuristics. An arithmetic representation where attributes might be represented as mathematical operators is one method[6], although rule-based representations are also possible [7].

Generation constructive hyper-heuristics are a powerful tool in the hyper-heuristic arsenal for providing general solutions to combinatorial problems. Existing work has primarily focused on the use of GP for this task but the potential remains for an ant-based application.

2.2. Ant Colony Optimisation

Ant algorithms are a broad class of algorithms that draw inspiration from behaviours of real-life ants [8]. Since its inception, ant-based algorithms have been successfully applied to a variety of problems, most notably combinatorial and search problems [9].

The heart of most ant algorithms is a cooperative interaction between all ants in the colony. This is facilitated through an indirect sharing of information about constructed solutions through a search space common to all ants [9]. As the colony continues to search the space there is a tendency towards the increase of pheromone in good areas of the search space which gradually produces better solutions. As this is a stochastic process, there are no guarantees that the found solution will be the best, especially in high dimensionality problems. An ant-

based algorithm will eventually converge although the speed of convergence is problem-dependent [10].

Ant algorithms have been applied to several different problem domains like the travelling salesman problem (TSP) [9], but their application has focused on directly applying the ant algorithm in the solution space of the problem. The focus of this research, by contrast, is on the application of ant-algorithms to search the heuristic space. This develops the field further as the exploration of the heuristic space by ant algorithms is not well researched, especially for generation constructive hyper-heuristics.

In terms of hyper-heuristics ant algorithms have been utilised in hyper-heuristics although their application has focused solely on selection hyper-heuristics [4]. An early example of the application of ant colonies to hyper-heuristics is the work of [11]. The authors describe a process by which an ant colony can be used to make choices about which heuristics to use to construct a solution for a 2D bin packing problem. This early example of a selective constructive hyper-heuristic demonstrated that ant-based methods can be used for hyper-heuristics but this has not been sufficiently investigated.

This leaves the potential, then, for applying ant-based techniques to generation constructive hyper-heuristics, where ant methods can be used to drive the creation of heuristics that can then be applied to problems in the underlying solution space.

In particular, existing work within hyper-heuristics has only focused on the pure application of an ant-based method to a given hyper-heuristic task. A comparative study of how the pheromone space, the central mechanism of an ant-based technique, is applied to hyper-heuristic tasks remains to be investigated.

3. Pheromone Spaces

A full explanation of pheromone spaces is provided in the supplementary material [13]. This section is therefore dedicated to explaining the necessary

modifications to the pheromone space that apply in the context of this research as well as their implications.

3.1. Pheromone Space Projection

As described in the supplementary material [13], moving the ant algorithm to the heuristic space necessitates (with a 2D pheromone map) a loss of information. 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 1.

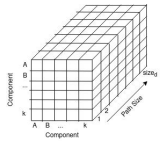

Figure 1: 3D Pheromone Map

In this new projection, the third dimension represents points in the heuristic path being created by the ant and is 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 links taken at different points in the search.

The limit of this third dimension is dependent on the maximum size of the path. In GP, a maximum tree depth is used to limit the size of the tree. With HACO, the limit is the number of operators allowed in a single path. Since the operators grow the path (by necessitating inputs of other components), the limit to a given path is the number of operators. From this, the size of the third dimension, $size_d$, is therefore calculated as:

$$size_d = f_{limit} * maxArity(f) \tag{1}$$

where f_{limit} represents the maximum number of operators allowed in a single path which is an algorithm parameter and *maxArity(f)* returns the largest arity value of any of the operators. Arity in this context refers to the number of input

arguments to a given operator. For example, the addition operator has an arity of 2 as it requires two inputs.

3.2. Effect on Searching

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 heuristics. 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.

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.

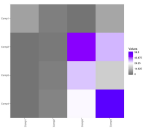

Figure 2: 2D Pheromone Map

The pheromone map in Figure 2 is represented with the pheromone concentrations given by a colour gradient.

In Figure 2, 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, those 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. 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 precise heuristic refinement is unnecessary, a 2D pheromone map would enable the ants to continue to do their local searching until they have 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 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 map 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 increase the search effort required to find good solutions because the space of potential searching has been magnified. 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. Heuristic Ant-Colony Optimisation

This section introduces the HACO algorithm and explains how an ant-based method is used as the search technique for a generation constructive hyper-heuristic.

4.1. High-Level Overview

The high-level algorithm of the HACO method is presented in Algorithm 1. This algorithm is a broad overview of functioning from iteration to iteration. Individual components and their functioning are detailed in their appropriate sections.

The logic of Algorithm 1 is essentially the same as a standard ant algorithm. A population of ants is initially created (with empty paths) between lines 2 and 3 and then over several iterations, they will construct paths, line 9. These paths are evaluated (with some fitness function (line 11)) and that information is used to update the shared pheromone map accordingly (lines 16 and 17). At the end of the algorithm's execution, the best solution found as well as the best path is returned. The pheromone map is initialised randomly with small random values in the range of [0,1] for the 2D and 3D HACO.

The design of the algorithm is primarily determined by considerations of the interaction between its ant algorithm basis and the adaptations needed to use the ant algorithm in the heuristic space. Ant algorithms are not typically employed in the heuristic space and as such, modifications are needed to enable the algorithm to operate in this new way. Specifically, adaptations are needed in the path construction and interpretation aspect of the ant algorithm as the path consists of low-level components in the heuristic space and not in the solution

Algorithm 1: 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      foreach a ∈ n_k do
8          Ant a constructs a path
9      foreach ant a in n_k do
10         fitness[a]=evaluate(a);
11         if fitness[a] < best then
12             best = fitness[a];
13             S_B=a.getSolution();
14             P_B=a.getPath();
15     evaporate ph using Equation 2;
16     update ph using Equation 3;
17     update p using Equation 4 and update α using Equation 5;
18     i = i + 1;
```

space. These adaptations also have to take into account the projection of a 2D pheromone map into a third dimension as well and how that affects ant navigation.

This high-level overview also represents the most basic methodology of the operation of an ant algorithm in terms of detailing broadly how the ant algo-

rithm will function within the context of the adaptation for use by a generation constructive hyper-heuristic.

4.2. Pheromone Updates and Evaporation

Once the ants have constructed their path, two updates need to occur in Algorithm 1, lines 16 and 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:

$$ph_{xyz} = (1 - p) * (ph_{xyz}) \tag{2}$$

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) [12]. The only modification is to the specific pheromone update value, $\Delta\tau^k_{xyz}$. This is given by:

$$\Delta\tau^k_{xyz} = \left\{ \begin{array}{ll} \frac{1}{f(x^k)*len(x^k)} & \text{if link (x,y,z)} \in \text{path } x^k \\ 0 & \text{if link (x,y,z)} \notin \text{path } x^k \end{array} \right\} \tag{3}$$

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.

4.3. Decay Function

There are two control variables (p and α) used in Algorithm 1. The former variable is used to control the rate of evaporation during the execution of the algorithm. 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 that updates these values after every iteration t. These are as follows:

$$p_t = (p_{init} - p_{final}) * \frac{it - t}{it} + p_{final} \tag{4}$$

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.

$$\alpha_t = (\alpha_{init} - \alpha_{final}) * \frac{it - t}{it} + \alpha_{final} \tag{5}$$

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

4.4. Path Construction

As ants traverse through the component space they will gradually add nodes to their path. This path has to be converted into a format that can then be interpreted as a heuristic. This process is detailed in the provided supplementary material [13].

In generation constructive hyper-heuristics the typical usage of the constructed heuristic is as a control function. 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 process starts 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. 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 thing that determines how large a heuristic can grow.

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 algorithm then adds components based on the arity of the first operator. This process will then defer to the compute function to convert a given node into a heuristic component that builds the solution over time. Finally, the last character is removed from the completed heuristic as this last character will be a terminating character like the semicolon.

4.4.1. 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 2. This algorithm is a recursive process whose initial function is set up by the path construction 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.

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.

The compute function, Algorithm 2, does the conversion of the path node into the string heuristic representation. The process is largely the same as the path construction process but with the addition of the return statement, line 12, which returns either the domain attribute, which does not enable additional

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

Figure 3: Examples of Expressions

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

additions to the path, or an operator. An operator requires inputs based on its arity value and these inputs necessitate adding new components to the path. 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.

4.5. Node Selection

Algorithm 3 describes the process of choosing nodes. 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 [14, 15]. In terms of the calculation, there are two factors in choosing a node: heuristic desirability, h, and pheromone concentration, ph.

The process calculates the amount of pheromone based on the node i moving to node j on the current 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.

The desirability heuristic, h, is simply:

$$h(x, path) = \frac{1}{count(x, path)} \tag{6}$$

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.

4.6. 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

Algorithm 3: Node Selection Process

Input: a ant

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

$set = \emptyset$;

if $a.currF < limit$ **then**

  $set = domainAttSet + operatorSet$;

else

  $set = domainAttSet$;

n_{ind}=indexOf(curr node of a.path);

l_{ind}=a.path.size()-1;

$tmp[] = [set.size()]$;

$phs[] = [set.size()]$;

$sum_{prob} = 0$;

for $j < set.size()$ **do**

  $tmp[i] = \alpha * ph[n_{ind}][j][l_{ind}] + (1 - \alpha) * h(node_j, a.path)$;

  $sum_{prob} += tmp[j]$;

for $i < set.size()$ **do**

  $pks[i] = \frac{tmp[i]}{sum_{prob}}$;

$ind = 0$;

$sum = pks[0]$;

$r = U(0, 1)$;

while $sum < r$ **do**

  $ind = ind + 1$;

  $sum = sum + pks[ind]$;

$choice = set[ind]$;

if $choice \in operatorSet$ **then**

  $a.currF += 1$;

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.

5. Hybridising the Pheromone Spaces

With the assumption that the different types of pheromone maps will have different advantages and disadvantages, it then follows that a hyper-heuristic technique that hybridises the use of two types of pheromone maps will result in an improved algorithm that would be more broadly applicable than individual algorithms with either pheromone map.

In some sense, the task of hybridisation is similar to the effect of an ensemble strategy [16]. In particular, an ensemble relies on multiple algorithms to facilitate a collective learning process. In this case, the different pheromone maps share information but do not run concurrently. Rather they simply share the information without using a consensus strategy.

Hence the development of a hybrid technique called heuristic ant-colony optimisation hybrid (HACOH). As the hybrid make use of both 2D and 3D pheromone maps, there is no need to specify the type of pheromone map when referring to the algorithm.

5.1. Hybridisation

The hybridisation is achieved through the use of two separate ants, with 2D and 3D pheromone maps respectively. A list is used to decide, during a run, when to execute an iteration with which type of ant (and therefore using which type of pheromone map). The size of the list is equal to the number of iterations allowed in a run, where each item is either a 2 or 3, indicating whether to do an iteration with the 2D pheromone map or the 3D pheromone map.

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

```
<2,2,2,3,3>
```

In the example, the first three iterations of the run are performed with the 2D pheromone map, and the remaining two are done with the 3D iteration map.

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 two ants share information about the results of their searches in their respective spaces with each type of ant contributing to an overall search of the entire space. The size of the structure 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.

5.2. List Optimisation

Given that the list can be arranged in many combinations, some kind of optimisation is needed to determine the optimal configuration of 2D and 3D iterations to produce the best list 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) for generating constructive heuristics for a given problem. This methodology is given in Figure 4.

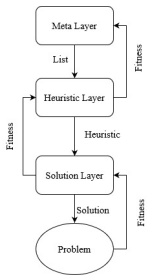

Figure 4: Model of Hybridisation

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 use of

two pheromone spaces is therefore a meta-optimisation task, to the underlying hyper-heuristic task.

5.3. Optimisation Strategy

The algorithm for the ILS optimisation strategy is provided in the supplementary material [13]. In the 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 several specified iterations, returning the best list as the solver. A memory of the generated lists is used to prevent reusing previously generated lists.

5.3.1. Perturbative Function

The *moveAccept* function is used to modify the existing list according to a perturbation rule that was created for this application. The algorithm is provided in the supplementary material [13].

The function returns a unique list not seen in the memory thus far. If the size of the list is under ten, 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. The perturbative operator is also provided in the supplementary material [13].

5.3.2. Move Acceptance

A *moveAccept* function is used to determine whether to update the list and thus whether or not the "new move", that is a new list, is accepted in the search. The full description of this process is detailed in the provided supplementary material [13].

5.4. List Approximation

One of the core 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 that 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 five $< 2, 2, 3, 3, 2 >$ that is being expanded to size thirteen. 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 within more reasonable boundaries.

6. Experimental Methodology

In this section, the experimental methodology used in this paper is presented. This will consist of the specific details of the problems, datasets and experiments.

6.1. Problem Domains

There are two problems considered in this paper. A description of the problem will be presented alongside descriptions of the benchmark data considered

for each domain. The purpose of the experiments is to evaluate the HACO and HACOH algorithms for two different domains. This is not, necessarily, to improve on the optimal solutions for these domains, especially in the case of the TSP where several non-heuristic solutions do exist.

6.1.1. 1D Bin Packing Problem

The 1D bin packing problem (1DBPP) concerns itself with the task of packing n number of items into x number of bins with all bins typically of the same capacity. There are two benchmark sets used in this paper described in Table 1.

Table 1: Benchmark Datasets for 1DBPP

Benchmark Set	Number of Instances	Source
Uniform	80	[17]
Hard	10	[18]

All datasets (and instances) considered here are chosen specifically to make comparisons with the literature [19]. The fitness function is based on the one presented by [20]. This is presented in Equation 7.

$$Fitness = 1 - \left(\frac{\sum_{i=1}^{n} (\frac{\sum_{j=1}^{m} v_j x_{ij}}{C})^2}{n} \right) \tag{7}$$

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 prioritises minimising the wasted space in each bin, favouring bins that are nearly full or full, and avoiding plateau issues that might arise if just the number of bins was used as the fitness value. This is an improvement as any two solutions could use the same number of bins but have differing levels of fullness with the one minimising wasted space being preferable. Equation 7 provides a convenient way of calculating this.

6.1.2. Travelling Salesperson Problem

The TSP is an NP-hard optimisation problem. The problem consists of finding the lowest cost closed tour (cycle) in a graph that starts and ends at a given node and visits all other cities only once [21]. Twenty-one TSP instances are taken from the TSPLIB [22]. The instances are described in the supplementary material [13].

6.2. Problem Components

This section describes the low-level components for each of the different domains. These are namely operators and domain attributes. The operators will be described separately as both domains largely share the same operators. The domain attributes are described in their sections below but for greater detail, the referenced literature should be referred to.

6.2.1. Operators

The 1DBPP and TSP domains have the +, -, * and / operators in common. The 1DBPP domain makes use of the absolute value (A) operator in addition while the TSP domain makes use of the modulus operator (%). All functions are protected. All of the operators except for (A) have an arity of 2. Arity refers to the number of inputs required for that function.

6.2.2. Domain Attributes

In terms of the 1DBPP, the domain attributes are taken from [23]. In their work, they 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.

The domain attributes for the TSP are taken from [21] and were partially derived from existing TSP heuristics. They are described in the supplementary material due to their quantity [13].

6.3. Solution Construction Process

Typically a generation constructive hyper-heuristic will evolve a control function representing a heuristic that guides a 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. For different problems, the desirability score represents different aspects of the problem, like the desirability to add a given vertex into a current path for example. The full details of each of the solution construction methods are detailed in the provided supplementary material [13].

6.4. Ant Algorithm Parameters

In terms of operational parameters that are used directly in the algorithm, there are two: α and p. The former determines the pheromone desirability used in Algorithm 3 and the latter indicates the rate of evaporation used in Equation 2. The scale of the experimental trials means that performing exact parameter tuning for each domain or benchmark is computationally infeasible. Instead, both parameters make use of a schedule to modify their values throughout the execution of the algorithm. The term schedule in this context refers to a formula that determines a range of possible values for the given parameter based on the progress of the algorithm's execution during the run.

The initial value of p is 0.1 and it will be linearly increased to the value of 0.9 using Equation 4. The reason for this is that this achieves an evaporation rate that promotes the trend of exploration to exploitation the most. Specifically, when the rate of evaporation is low, the pheromone map will be saturated with pheromone. As time goes on and the rate of evaporation increases, this will have a filter effect on the pheromone map, eliminating all but the best concentrations of pheromone. Thus initially the algorithm will explore the space before gradually moving towards exploiting it.

The value α runs on a similar schedule using Equation 5. It starts from 0.1 before gradually moving to 0.9. Pheromone desirability is used to weigh the influence that the novelty heuristic has in the node selection process as compared

to pheromone. The desirability of pheromone is α and the desirability of the heuristic is $(1-\alpha)$. Thus initially the heuristic will be favoured before a gradual shift to the value of the pheromone.

This is important because the novelty heuristic will bias towards solutions that are as diverse as possible, leading to greater exploration of the heuristic search space. As the weight associated with the heuristic declines, the influence of the pheromone increases. This corresponds to the increased evaporation rate, meaning increasingly only the better paths are surviving more evaporation, enabling better exploitation. Thus these two schedules work in concert to deliver a reasonable parameter set for the algorithm during its execution across all problem domains.

The schedules employed for these parameters have been chosen because they facilitate general behavioural trends with regard to the algorithms. The point of this research is to do comparisons of ant algorithms with different pheromone maps as they drive hyper-heuristics. Therefore generalised parameter schedules that work for all the problems are preferable to specific parameter choices that might yield optimal values for some cases and not others. This configuration will be used for 2D and 3D HACO algorithms and reflects a parameter tuning strategy.

6.4.1. Generating New Lists

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, p_l. The value of n_k is 10, the number of runs is 30, the number of iterations (ILS) is 10, the number of iterations (AS) is 30 and the path length p_l is 10.

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.

In addition to the reduced parameters, the process here only focuses on a subset of the full benchmark sets for each problem. For the 1DBPP domain, fifteen instances were randomly selected, 3 from each instance type (based on the problem size), to make a subset of the data. For the TSP domain, seven instances were chosen based on their size. These are bier127, d493, d657, eil51, fl1577, kroA150, and u724. This gives a moderately sized set of data of various problem sizes to generate lists from.

The generated lists will then be expanded for each instance in the full benchmark set for both problem domains. 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.

6.4.2. Iterated Local Search Parameters

The ILS algorithm has a parameter specific to it: *mp*. This parameter is the rate of perturbation and it decides 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 and 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.

6.5. Experiments

In terms of this paper, there are two principle experiments to be conducted. The first is to perform experiments with the HACO algorithm. These are executions of the HACO algorithm using 2D and 3D pheromone maps respectively on the TSP and 1DBPP domains. The second experiment involves the same domains but with the HACOH algorithm instead.

These experiments are conducted to determine the differences between the different kinds of pheromone maps (2D vs 3D) and to determine if the hybrid algorithm provides any benefits. The experimental parameters and conditions are specified in their sections below.

6.5.1. Experimental Parameter Tuning

In terms of the important parameters for Experiment 1, the number of ants n_k and the number of iterations are the most important. The number of runs is fixed at 30 to provide a large sample of data for statistical testing. To determine good values for the other two parameters a grid search procedure was used to test the performance of the 2D HACO algorithm under different configurations of n_k and the number of iterations.

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. Hence the 2D HACO algorithm is used for parameter tuning and the chosen parameters will also be applied to the 3D HACO during the experiments.

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. 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. As the purpose of this parameter tuning is to establish good values for the comparison of algorithms, and not merely to produce good values, these ranges are more than sufficient for this research.

In terms of the important parameters for Experiment 1, the number of ants n_k and the number of iterations are most important. The number of runs is fixed at 30 to provide a large sample of data for statistical testing. To determine good values for the other two parameters, a grid search procedure was used to test the performance of the HACO algorithm under different configurations of n_k and the number of iterations. The path limit variable p_l is fixed at 10 as this value results in heuristics that are not large enough to be unreadable by a person whilst being large enough to facilitate some degree of complexity.

With these values, a matrix of the different combinations was set up and the HACO algorithm was run under each different configuration. The average of the best fitness values over the number of runs was taken to assess each configuration. This tuning was done with the u120 and berlin52 instances from the 1DBPP and TSP benchmark sets respectively. These parameters were chosen based on their complexity. They represent small enough problems to extensively parameter tune with whilst being large enough to not be entirely trivial to solve.

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, the point of this testing is to establish a baseline understanding of how the algorithm performs under different configurations in the respective domains. These smaller scale tests are meant to provide rough approximations of the algorithm performance so that larger and more comprehensive tests can be performed.

6.5.2. Experiment 1: 2D vs 3D Pheromone Map Comparison

Figures 5 and 6 present the result of the parameter tuning process as described in Section 6.5.1. Pearson's Correlation Coefficient [24] is also calculated for the relationship between fitness and n_k and the number of iterations respectively.

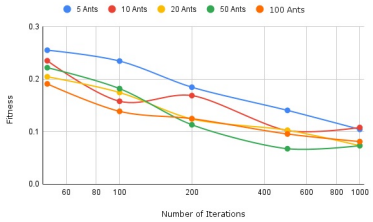

Figure 5: Fitness Results of 1DBPP Parameter Tuning Process

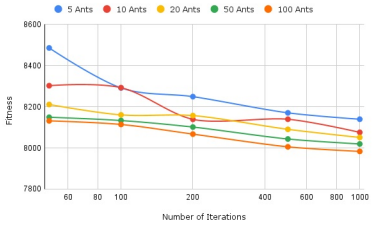

Figure 6: Fitness Results of TSP Parameter Tuning Process

Table 2: Correlation Coefficients between Fitness, n_k and Number of Iterations

Correlation Coefficient	1DBPP	TSP
n_k-Fitness	-0.279	-0.573
Iterations-Fitness	-0.770	-0.595

In terms of the results, Figures 5 and 6 both show strong trends with regards to better fitness values being assessed as the number of iterations and n_k increase. In terms of the correlations, Table 2, there are negative correlations in all comparisons and domains. This indicates that as the experimental parameters increase, there is a tendency in the fitness to go down.

However, for the 1DBPP domain, the correlation between the number of ants, n_k is a weak negative correlation at only -0.27. Whereas the correlation for the number of iterations is much stronger at -0.76. This would suggest that more iterations are more important in determining the quality of the fitness than simply adding more ants. For the TSP domain, the correlations are both negative but much closer together at around -0.58. This would suggest that the number of ants and number of iterations are equally important. That these trends hold for both domains would indicate that there is validity to the nature of the algorithm's performance concerning its experimental parameters.

From the results presented above, the parameters listed below will be used for the experimental comparison between HACO algorithms using the 2D pheromone and 3D pheromone maps respectively. These parameters are n_k, the number of ants, the number of runs, the number of iterations per run, and the path limit, p_l. The parameters for the first experiment are:

- Number of Runs: 30
- p_l: 10
- Number of Iterations: 300
- n_k: 10

The first two parameters were chosen for reasons discussed in Section 6.5.1. The next two parameters' values were chosen based on a compromise between computational effort and algorithm performance. Specifically, after the 200 iteration mark, the degree of improvement as the number of iterations starts to increase, decreases rapidly. The choice of 300 iterations is a good compromise as it still enables better performance than prior values, but is far less computationally expensive than the larger options.

For the choice of n_k, there are some instructive insights from the results. Firstly, the number of ants is not as important for the 1DBPP domain as the number of iterations so choosing fewer ants but at a higher number of iterations would not degrade the performance too much. Secondly, in the TSP domain,

the performance of 10 ants is reasonably close to the larger ant values. Hence, the choice of 10 for n_k. This will result in meaningful performance without an excessive computational burden.

6.5.3. Experiment 2: Hybrid vs Non-Hybrid Comparison

The parameters listed below will be used for the experimental comparison between HACO and the HACOH algorithm. These parameters are n_k, the number of ants, the number of runs, the number of iterations per run, and the path limit, p_l. The parameters for the first experiment are:

- n_k: 10
- Number of Runs: 30
- Number of Iterations: 300
- p_l: 10

The parameter values listed here were chosen based on the experimental testing procedures that were employed in Section 6.5.1. Mirroring the parameter values in the second test also ensures a fair comparison can be made between the HACO algorithms used by the HACOH and the HACO algorithms on their own, allowing for the isolation of the effect of the HACOH algorithm to the overall hyper-heuristic process.

Using the list approximation method described in section 5.4, the HACOH algorithm is run under the same conditions as the HACO variants. The lists are generated based on the process described in section 5.4. The purpose of this experiment is to therefore test the performance of the hybridisation under similar conditions to the non-hybrid forms. While not perfect, the approximated list should be closer in form whilst still allowing the experiments to complete within a reasonable time.

6.6. Assessment Metrics

In terms of assessment metrics, each problem will be assessed according to the practices typical of that problem.

6.6.1. 1DBPP

The 1DBPP will require a different approach due to the number of instances involved in the benchmark sets. In particular, the calculation of a statistic called the Ratio [19]. This formula is provided in the supplementary material[13].

6.6.2. TSP

The TSP assessment will consist of presenting the average performance over all of the runs. This is formalised by the following equation:

$$\frac{\sum_{i=1}^{n_r} F_r}{n_r} \tag{8}$$

In Equation 8, F_r is the fitness after a run of the algorithm and n_r is the number of runs of the algorithm.

6.6.3. SDD

To assess the various algorithms in terms of their generality, a generality metric, standard deviation of distances or SDD, is employed [25]. It is formulated as:

$$SDD(H) = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}} \tag{9}$$

SDD is designed to assess the performance of a hyper-heuristic over 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 (2D, 3D or hybrid) will be able to generalise across problems and domains. This is particularly important for the hybrid as a primary function of the hybridisation is to improve the algorithm's generality by incorporating both kinds of pheromone maps.

6.6.4. Comparison with Existing Methods

The primary focus of this work is to assess the effect that different pheromone maps have when used in ant-based hyper-heuristics. To that end, the primary axis of the analysis will be centred on the comparison of the 2D HACO, 3D HACO, and HACOH in the respective problem domains. However, it is still

important to contextualise the results that are produced within the wider field of the problem domain.

To that end, existing construction heuristics from within the TSP and 1DBPP domain will be presented alongside the HACO and HACOH results to contextualise the performance of the algorithms in terms of how well the generated heuristics can compete against existing good heuristics in the field. Additionally, a comparison GP algorithm from the TSP and 1DBPP domain will be provided as well to demonstrate the capacity of the HACO and HACOH algorithms against a hyper-heuristic. The goal of this comparison is to contextualise the performance of the algorithms in their field, and not to do a pure comparison.

For the 1DBPP domain, the following heuristics will be used for comparison: First Fit (FF), Best Fit (BF), Next Fit (NF), Worst Fit (WF), First Fit Decreasing (FFD), Best Fit Decreasing (BFD) and Next Fit Decreasing (NFD).

These heuristics were chosen as they are widely used [2]. The comparison GP for the 1DBPP domain was taken from [23]. Except for the GP (whose results are taken from literature), the heuristics were implemented for the experiments.

For the TSP part of the experiment, the following heuristics will be used for comparison: Nearest Neighbour, Nearest Insertion Greedy and Christofides. A TSP GP is compared as well.

These were taken specifically for comparison from existing recent literature [26]. It also bears noting that the TSP problem, in particular, has had significant development [27] as a problem domain and that there are exact methods that exist that are capable of optimally solving TSP instances. An example of this would be the Concorde Solver which is a proven and effective exact TSP solver [28]. As this is the case, the optimal values found by these exact techniques for each instance will be compared with the results of the HACO and HACOH algorithms to contextualise the results of the ant-based hyper-heuristics.

6.6.5. Statistical Testing

To properly assess the validity of the comparisons between the different ant-based hyper-heuristic methods, a statistical testing procedure is applied. Firstly the three algorithms (2D HACO, 3D HACO, and HACOH) will be assessed using Friedman's Test [29] to assess if there are meaningful statistical differences between the different algorithms. Then afterwards a post-hoc analysis will be conducted using the Mann-Whitney U Test [30]. The tests will be one-tailed tests conducted at a 0.05 level of significance.

- 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 standardised effect size will be included alongside the results of the statistical tests. This metric quantifies the magnitude of the difference between two techniques' results in terms of a value in the range of [0,1]. The larger the effect size, the larger the magnitude of difference between the two techniques.

6.7. Technical Specifications

For this research, a computing cluster provided by the University of Pretoria was used. 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.

7. Results and Discussion

This section presents the results of the experiments. Discussion and interpretation of the results will be provided alongside the results as well. Information about the fitness values can be found in Section 6.1.

7.1. 1DBPP Results

In Table 3 the results of the HACO and HACOH experiments are given. The comparison methods are also provided for brevity in the table. The best result is indicated in bold. The heuristics used in this comparison are detailed in Section 6.6.4.

Table 3: 1DBPP Results by Benchmark

Method	Uniform		Hard	
	Ratio	Std Dev	Ratio	Std Dev
2D HACO	1.0558	0.0180	1.0792	0.0063
3D HACO	1.0253	0.0077	1.0624	0.0088
HACOH	1.0352	0.0104	1.0679	0.0117
First Fit	1.0586	0.0113	1.0606	0.0099
Best Fit	1.0545	0.0108	1.0606	0.0099
Next Fit	1.3117	0.0175	1.1566	0.0117
Worst Fit	1.1477	0.0200	1.0606	0.0099
First Fit Decreasing	1.0129	0.0062	1.0606	0.0099
Best Fit Decreasing	1.0129	0.0062	1.0606	0.0099
Next Fit Decreasing	1.4012	0.0195	1.1566	0.0117
GP [23]	**1.0000**	0.0003	**1.0004**	0.0070

Table 3 breaks down the results by the benchmark set for the 1DBPP domain. In addition to the ratio value, a standard deviation is given as well to indicate the degree of variance of each method. In terms of the outcomes, the best performing method overall is the GP-based method with the best performing ant-based hyper-heuristic being the 3D HACO for both benchmarks.

7.1.1. Comparison of HACO and HACOH Algorithms

The tables for the statistical testing for the comparison between the 2D HACO, 3D HACO and the HACOH have been provided in the supplementary material [13].

In terms of the results of the Friedman test, the test finds a statistically significant difference between the three algorithms with a small p value and large χ^2 value. This suggests that the different pheromone maps played a significant role in affecting the performance of the different ant-based hyper-heuristic algorithms.

The results of the post-hoc analysis compared the 2D HACO against the 3D HACO and the HACOH against the 2D and 3D HACO respectively. In terms of the results, the first test results in not rejecting H0. Based on the large standardised effect size, which quantifies the degree of difference between the two groups, the 3D HACO is significantly better than the 2D HACO in the 1DBPP domain.

The remaining two tests assess the differences between the HACOH and the non-hybrid algorithms. H0 is rejected in the comparison with the 2D HACO and not rejected in the comparison with the 3D HACO. From this, it is apparent that the HACOH algorithm is better than the 2D HACO for the 1DBPP but not better than the 3D HACO. The standardised effect sizes are also smaller than in the first comparison, at 0.53 and 0.4 respectively. The outcome of this assessment is a clear indication that the 3D HACO is superior to the HACOH and 2D HACO for this problem domain.

7.1.2. Contextualisation with Existing Methods and Construction Heuristics

While not the primary focus of this work, the comparisons with other existing methods (construction heuristics and GP) have yielded some interesting insights. Firstly, the best performing method overall is the GP and this reflects the prevalence of GP for generation constructive hyper-heuristics, so it having the best results should not be a surprise.

However, the 3D HACO has admirable results in comparison to the remaining construction heuristics. It outperforms five of the seven construction heuristics in the uniform benchmark by a relatively wide margin (FF, BF, NF, WF, and NFD). In terms of quantifying the differences between the FFD and BFD, the standardised effect size is about 0.32 between the 3D HACO and the

best performing construction heuristics. This is a magnitude of medium to low size indicating that the differences do exist in a meaningful sense but that they are not entirely separated in terms of performance.

7.2. TSP Results

Due to their size, the TSP results tables are provided in the supplementary material[13]. Tables 7 and 8, in the supplementary material, provide the results of the HACO and HACOH experiments for the TSP domain with the comparison results. The average for all of the instances and the corresponding standard deviation are provided as well. The comparison includes the construction heuristics as well as a TSP GP method. In terms of these results, the Christofides heuristic does the best in the most number of instances with the second-best being the 2D HACO.

7.2.1. Comparison of HACO and HACOH Algorithms

The tables for the statistical testing for the comparison between the 2D HACO, 3D HACO and the HACOH have been provided in the supplementary material [13]. The results of the Friedman Test indicate that the differences between the three groups (2D HACO, 3D HACO, and HACOH) are significant enough to be statistically meaningful.

The results of the post-hoc analysis compared the 2D HACO against the 3D HACO and the HACOH against the 2D and 3D HACO respectively. In terms of the results, the comparison between the 2D and 3D HACO resulted in not rejecting H0. Given that the standardised effect size was very small, 0.0097, this indicates that the two groups (2D and 3D HACO) are very close together in performance and more likely to be equal.

The implication of not rejecting H0 for the comparisons between the 2D HACO, 3D HACO, and HACOH are different, however. Given their larger standardised effect sizes, 0.12, not rejecting H0 indicates that the HACOH algorithm produced worse fitness outcomes than the 2D and 3D HACO respectively. This can be further corroborated by the average values over all the instances in the

TSP domain. The HACOH algorithm performed worse in this domain than the other two with the differences between the 2D HACO and 3D HACO being very marginal.

7.2.2. Contextualisation with Existing Methods and Construction Heuristics

Although it is not the primary aim of this work, contextualising the HACO and HACOH algorithm performance against existing construction heuristics and recent methods in the TSP domain can yield some valuable insights. Of course, exact solutions for the TSP exist, such as Concorde, and therefore the best values for the TSP instances are included in Tables 7 and 8.

The comparison showed that in general, the Christofides construction heuristic outperformed most of the other methods across the benchmark set. There were some cases where the 2D HACO and 3D HACO performed best, however. These were bier127, kroC100, pr226, ts225 for the 2D HACO, and eil51, lin318, and pr264. In terms of the performance, the 2D and 3D HACO algorithms outperformed all of the other construction heuristics and even the TSP GP in this domain. It is unusual given the dominance of the GP in the 1DBPP domain but this may be indicative of the different nature of the problem domain. The TSP is a domain that is far closer to the domains that ant algorithms are used in as opposed to the 1DBPP.

In terms of the degree of difference between the best performing HACO, the 2D HACO, and the best performing method, the Christofides heuristic, the standardised effect size is only 0.031. As the standardised effect size indicates the magnitude of difference between the two methods, a value of 0.031 is a very small difference between them. This indicates that the HACO algorithm can deliver comparable performance to the best performing methods in this domain.

7.3. Generality

The SDD score is an important measure of assessing the generality of a given hyper-heuristic. The SDD score is a metric taken from existing literature [25]. It is used to assess the generality of a given hyper-heuristic.

In hyper-heuristics, generality is an especially important metric because it enables hyper-heuristics to be widely applicable across different domains or benchmark sets within a domain. The most successful solution in terms of its raw results is often not one that can be broadly applied, hence why hyper-heuristics are considered; they can generalize better by performing well on different problem instances instead of producing good results for some problem instances and poor results for others. Therefore looking at the generality of these hyper-heuristics is important to be able to assess their value.

Table 4: Summary of SDD Scores for 1DBPP and TSP Domains

SDD Score	2D HACO	3D HACO	HACOH
1DBPP	1.752	1.351	1.408
TSP	3.676	3.765	3.555
Average	2.714	2.558	2.482

In Table 4 the SDD scores for all of the hyper-heuristics are presented for each domain. This level of generality is presented for the problem domains specifically as this is most important to the analysis.

The results show that in terms of the non-hybrid hyper-heuristics, the 3D HACO is better than the 2D HACO in terms of generality for the 1DBPP domain with the situation reversed for the TSP domain. The HACOH algorithm has the lowest SDD score for the TSP domain and a score for the 1DBPP domain that is only slightly higher, 0.057 higher, than the 3D HACO. When the average of the SDD score is calculated, the HACOH algorithm emerges with the best generality score across both domains with the 3D HACO in second place.

7.3.1. Interpretation

The SDD score highlights the comparison between the different hyper-heuristic algorithms. Based on these results it is the case that the HACOH algorithm generalises the best across both domains. On average, its SDD score is the lowest whereas, inside a single domain, a non-hybrid HACO algorithm might

do better, no algorithm does as well as the HACOH across both domains.

Using a single HACO algorithm for a given domain could theoretically result in a better performance for the hybrid, but the hybrid can perform across many domains where the non-hybrids might be insufficient. The list generation process itself is a computationally expensive operation and so this has to be compared against the cost of selecting the best pheromone map for the given problem.

7.4. Pheromone Maps

The pheromone maps and their analysis are presented in the supplementary material as this analysis is of secondary importance to the overall research [13].

7.5. Runtimes

Part of the process of comparing the different algorithms will be to assess the runtimes of the algorithms in the different domains. This is not a wholly objective measurement as different hardware and operating environments can affect the final runtimes of the algorithms, but it is added for the study's completeness. The tables showing the runtimes have been provided in supplementary material [13].

In general, the 3D HACO takes the longest time over the different instances to execute a single run, except in the case of instances from u120, where HACOH takes the longest. As the scale of the problem increases, the time taken for a single run also increases. The increased time of the 3D HACO should be contrasted against the far better results that it produces in the 1DBPP domain.

There are two observations to be made with regards to TSP runtimes. The first is that the HACOH algorithm has faster times on the larger problems such as rl1889 and u1817 with the margins between the algorithms closing on the smaller problems like eil51 and ch130. The second observation is that the times of the 2D HACO and 3D HACO are incredibly close together.

While the HACOH does have better runtimes than either the 2D or 3D HACO, it has worse performance and so that has to be considered with regard to the runtimes. The more interesting observation is that the runtimes of the 2D

and 3D HACO do not differ significantly. The standardised effect size between them is 0.0058 which indicates a very narrow margin of difference. This is in sharp contrast to the difference between the 2D HACO and 3D HACO in the 1DBPP where the 3D HACO took longer on average with its runtimes.

7.6. Discussion

This section provides a discussion of the results in their totality, with analysis and insight provided as well. The aim is to contextualise the results in the wider context of the research and its aims.

7.6.1. 2D vs 3D HACO

The primary aim of this research is to examine the effect that using different (2D and 3D) pheromone maps will have when used in an ant-based generation constructive hyper-heuristic. The hypothesis that underlies this research is that different pheromone maps will have different effects on the ant-based hyper-heuristics that make use of them. The results of the experiments described in Section 6 are presented in this section. In their totality, the results demonstrate some important things regarding the use of 2D and 3D pheromone maps in the ant-based hyper-heuristic described here.

Firstly, the type of pheromone map is extremely important for the 1DBPP domain. There is clear statistical evidence presented in Section 7.1 that the 3D HACO outperformed the 2D HACO in the 1DBPP domain. It did so by a wide margin, albeit at an increased runtime cost. This highlights an important insight into how pheromone maps should be considered for this problem. Namely, 3D pheromone maps offer improved performance but will require more runtime to succeed.

However, as the results of Section 7.2 demonstrate, it is not a universal fact of the application of the 3D pheromone map. The results of the TSP comparison show that the 2D and 3D HACO algorithms have very similar performance metrics in both optimality and runtime. The implication, of course, is that

there is no appreciable difference in using a 2D or 3D pheromone map for an ant-based hyper-heuristic in the TSP domain.

There are many reasons why this may be. The TSP domain does not have as many constraints as the 1DBPP and the heuristic landscape may be simple enough to traverse effectively with both a 2D and 3D pheromone map. The telling indication is that despite being more complicated in structure and size, the 3D HACO has a similar runtime to the 2D HACO in the TSP benchmark. The more complicated boundaries found in the 1DBPP heuristic space are better navigated by the 3D pheromone map which can represent more information than the 2D pheromone map. However, when the landscape is less constrained, like, in the TSP domain, the difference is less meaningful.

The other factor to consider is the generality of the 2D and 3D HACO. The SDD scores of the 2D and 3D HACO indicate that for the TSP domain, the 2D HACO provides better generality, 3.676, as opposed to the 3D HACO at 3.765. Whereas in the 1DBPP, the situation is reversed with the 2D HACO having an SDD score of 1.752, and the 3D HACO having an SDD score of 1.351 which is better. Generality and optimality are not necessarily the same and these two factors can be at odds with one another. An algorithm that is more generalisable, may have issues with producing more optimal outcomes. With this method of assessment, the 2D HACO is preferable for the TSP domain as it provides better generality and the 3D HACO is better for the 1DBPP domain as it provides better generality.

The TSP and 1DBPP are very different but common discrete combinatorial problems and their inclusion in this research is used to provide wide coverage of types of combinatorial problems for the HACO algorithm to execute. The nature of the differences between the two types of pheromone maps largely relates to the core objectives of any hyper-heuristic: optimality and generality. In this regard, the 2D pheromone map is capable of providing better generality for the TSP domain and the 3D pheromone map is capable of providing better optimality for the 1DBPP domain. Both types of maps have strengths and weaknesses and this demonstrates that different pheromone maps can be significant to the

performance of ant-based hyper-heuristics but not that there is a universally preferable pheromone map for all types of problems or hyper-heuristic needs. That is, this research has demonstrated the no-free lunch theorem for the use of different pheromone maps for ant-based hyper-heuristics.

7.6.2. Hybridisation

The other aim of this research is to examine the effect that hybridising the pheromone maps has on the performance of ant-based generation constructive hyper-heuristics. In this regard, the amount of information gleaned from the results of the experiments is more than sufficient to provide insights into the effectiveness of the HACOH.

Firstly, the hope of the hybridisation was that the combination of pheromone maps would outperform the non-hybrid versions of the HACO algorithm. In this regard, the available evidence strongly disproves this claim. In the 1DBPP domain, the HACOH was only marginally better than the 2D HACO and far below the performance of the 3D HACO. In the TSP domain, the HACOH underperformed against both the 2D and 3D HACOH in all instances. Therefore, from the available evidence, the HACOH pheromone map hybridisation algorithm does not result in meaningful improvements over non-hybrid HACO algorithms in terms of optimality of results in the examined domains.

With regards to generality, the other main aspect of a hyper-heuristic, the HACOH algorithm achieved the best SDD score, 2.4815, on average over all the domains. However, this is contrasted against the score of the 3D HACO which differs by 0.0765, a small amount. The generality to be gained by using the hybrid HACOH algorithm is not enough to offset the loss in optimality incurred by using it.

In terms of factors that explain why the hybridisation of pheromone maps failed to achieve success, the most immediate answer lies in the hybridisation method itself. The hybridisation algorithm generates the list that determines how the different algorithms using 2D and 3D pheromones will be interleaved in terms of their search efforts. However, this interleaving is a static process that

is determined before executing the algorithm on the totality of the dataset, via a generation process. During each iteration, a different algorithm is executed and the results of its search are shared with the other algorithm.

The issue is that the search process is made more dynamic by the inclusion of multiple different pheromone maps and gains an additional dimensionality that it did not have before. Furthermore, while some information is shared amongst the different pheromone maps, this is only a partial reflection of all of the pheromone map information contained in the separate algorithms which can obscure vital information that the ant algorithms may need for their searches. It may also be the case that the different dimensions of the different pheromone maps cannot incorporate information from other types of pheromone maps without losing the meaning contained within.

The hybridisation method failed to achieve its desired outcomes but this does not mean that the goal of utilising different pheromone maps is a lost one. Rather an alternative to the hybridisation method used here would be to separate the use of pheromone maps further and rely on something like an ensemble protocol to determine how multiple ant algorithms with their pheromone maps should search the heuristic space.

8. Conclusion

At the core of this paper is a question about how ACO could be improved for the task of hyper-heuristics. More specifically, this question was investigated in terms of 2D and 3D pheromone maps in use for generation constructive hyper-heuristics. The investigation presented a novel ant-based hyper-heuristic, the HACO algorithm, and studied the effect of the 2D and 3D pheromone map (2D HACO and 3D HACO) on this hyper-heuristic for two separate combinatorial optimisation domains, the TSP and 1DBPP.

The results of the various experiments have demonstrated that there are meaningful differences between the ways that different types of pheromone maps can be utilised for hyper-heuristics. In particular that the 3D HACO showed

better performance results than the 2D HACO for the 1DBPP domain. In the TSP domain, the results were much closer with a slight advantage going towards the 2D HACO. These results extended to the generality assessment where the 3D HACO showed better generality performance than the 2D HACO on the 1DBPP domain but vice versa on the TSP domain.

An important part of this investigation included examining how the two different pheromone maps could be hybridised in an algorithm (HACOH) and what that hybridisation would mean for the performance and generality. In this regard, the experiments showed that the HACOH achieved either middling performance results on average whilst also having the best generality on average, in terms of both domains.

These indications when taken together demonstrate that there is a utility to considering different kinds of pheromone maps for different problems but that the attempts to combine these pheromone maps into a single algorithm that improves on both performance and generality still requires further research.

Previously ACO was limited in how it could be applied to problems as it required specific solution representations. This paper demonstrates that through the utilisation of heuristic spaces, ACO-based techniques can be applied to any problem through a hyper-heuristic with little modification of the underlying operation of the ACO technique.

Future work, therefore, includes extending the HACO algorithm to other kinds of hyper-heuristics, like generation perturbative, and exploring and expanding the hybridisation approach with other techniques or exploring algorithm selection methods for deciding when to use which type of pheromone map. Additionally, a comprehensive study of the space and computational complexity of the HACO algorithm remains open for future research as further study into the algorithm progresses past this initial point.

Furthermore, there remains potential for the exploration of an ensemble technique (or several ensemble techniques) as an alternative to the hybridisation process described here, specifically an ensemble of different ant colonies with different pheromone maps. This need not be limited to purely the ant-based

hyper-heuristic put forward in this research. Rather an ensemble technique could combine the efforts of the ant-based hyper-heuristic with other hyper-heuristics for a hyper-heuristic ensemble.

References

[1] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, S. Schulenburg, Hyper-Heuristics: An Emerging Direction in Modern Search Technology, Springer US, Boston, MA, 2003. doi:10.1007/0-306-48056-5_16.

[2] N. Pillay, R. Qu, Hyper-Heuristics: Theory and Applications, Springer, 2018.

[3] E. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A survey of the state of the art, Journal of the Operational Research Society 64 (2013) 1695–1724.

[4] J. Drake, A. Kheiri, E. Özcan, E. Burke, Recent advances in selection hyper-heuristics, European Journal of Operational Research.

[5] J. Branke, S. Nguyen, C. Pickardt, M. Zhang, Automated design of production scheduling heuristics: A review, IEEE Transactions on Evolutionary Computation 20.

[6] E. Burke, M. Hyde, G. Kendall, J. Woodward, Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one, Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference (2007) 1559–1565.

[7] M. Bader-El-Den, R. Poli, S. Fatima, Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework, Memetic Computing 1 (2009) 205–219.

[8] M. Dorigo, G. Di Caro, Ant colony optimization: A new meta-heuristic, IEEE. 2 (1999) 1477 Vol. 2.

[9] L. M. Gambardella, M. Dorigo, Solving symmetric and asymmetric tsps by ant colonies, in: Proceedings of IEEE International Conference on Evolutionary Computation, 1996, pp. 622–627.

[10] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: A critical survey, Annals of Operations Research 131 (2004) 373–.

[11] A. Cuesta-Cañada, L. Garrido, H. Terashima-Marín, Building hyper-heuristics through ant colony optimization for the 2d bin packing problem, in: Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems - Volume Part IV, KES'05, Springer-Verlag, Berlin, Heidelberg, 2005, p. 654–660. doi:10.1007/11554028_91.

[12] M. Dorigo, Optimization, learning and natural algorithms, Ph.D. thesis, Politecnico di Milano (1992).

[13] E. Singh, N. Pillay, A study of ant-based pheromone spaces for generation constructive hyper-heuristics supplement material (2022).
URL https://drive.google.com/file/d/19piVUxhPozdSskkugc5qetwg0a15Lk4M/view?usp=sharing

[14] A. Lipowski, D. Lipowska, Roulette-wfheel selection via stochastic acceptance, Physica A: Statistical Mechanics and its Applications 391 (6) (2012) 2193–2196.

[15] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT Press, 2010.

[16] G. Wu, R. Mallipeddi, P. N. Suganthan, Ensemble strategies for population-based optimization algorithms – a survey, Swarm and Evolutionary Computation 44 (2019) 695–711. doi:10.1016/j.swevo.2018.08.015.

[17] E. Falkenauer, A hybrid grouping genetic algorithm for bin packing, Journal of Heuristics 2 (1) (1996) 5–30. doi:10.1007/bf00226291.

[18] A. Scholl, R. Klein, C. Jürgens, Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem, Computers & Operations Research 24 (7) (1997) 627–645. doi:10.1016/s0305-0548(96)00082-2.

[19] E. Burke, M. Hyde, G. Kendall, J. Woodward, Automating the packing heuristic design process with genetic programming, Evolutionary computation 20 (2011) 63–89. doi:10.1162/EVCO_a_00044.

[20] E. Falkenauer, A. Delchambre, A genetic algorithm for bin packing and line balancing, Proceedings 1992 IEEE International Conference on Robotics and Automationdoi:10.1109/robot.1992.220088.

[21] G. Duflo, E. Kieffer, M. R. Brust, G. Danoy, P. Bouvry, A gp hyper-heuristic approach for generating tsp heuristics, 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)doi:10.1109/ipdpsw.2019.00094.

[22] G. Reinelt, Tsplib—a traveling salesman problem library, ORSA Journal on Computing 3 (4) (1991) 376–384. doi:10.1287/ijoc.3.4.376.

[23] E. Burke, M. Hyde, G. Kendall, Evolving bin packing heuristics with genetic programming, in: Parallel Problem Solving from Nature - PPSN IX, 2006, pp. 860–869.

[24] K. Pearson, Note on Regression and Inheritance in the Case of Two Parents, Proceedings of the Royal Society of London Series I 58 (1895) 240–242.

[25] N. Pillay, R. Qu, Assessing hyper-heuristic performance, Journal of the Operational Research Society.

[26] G. Duflo, E. Kieffer, M. R. Brust, G. Danoy, P. Bouvry, A gp hyper-heuristic approach for generating tsp heuristics, in: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2019, pp. 521–529. doi:10.1109/IPDPSW.2019.00094.

[27] D. L. Applegate, The traveling salesman problem: a computational story, Princeton Univ. Press, 2007.

[28] D. Applegate, W. Cook, S. Dash, A. Rohe, Solution of a min-max vehicle routing problem, INFORMS Journal on Computing 14 (2) (2002) 132–143. doi:10.1287/ijoc.14.2.132.118.

[29] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, Journal of the American Statistical Association 32 (200) (1937) 675–701. doi:10.1080/01621459.1937.10503522.

[30] H. B. Mann, D. R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Statist. 18 (1) (1947) 50–60.