Solution Biases and Pheromone Representation Selection in Ant Colony Optimisation

Erin James Montgomery
BInfTech (Hons)

A dissertation submitted in fulfilment of the requirements of the degree of Doctor of Philosophy for the School of Information Technology, Bond University

14 June 2005

Statement of Originality

The material in this thesis has not been previously submitted for a degree or diploma in any university. To the best of my knowledge this thesis contains no material previously published or written by another person except where due acknowledgement is made in the thesis itself.

E. James Montgomery

Date:

Summary

Combinatorial optimisation problems (COPs) pervade human society: scheduling, design, layout, distribution, timetabling, resource allocation and project management all feature problems where the solution is some combination of elements, the overall value of which needs to be either maximised or minimised (i.e., optimised), typically subject to a number of constraints. Thus, techniques to efficiently solve such problems are an important area of research. A popular group of optimisation algorithms are the metaheuristics, approaches that specify how to search the space of solutions in a problem independent way so that high quality solutions are likely to result in a reasonable amount of computational time. Although metaheuristic algorithms are specified in a problem independent manner, they must be tailored to suit each particular problem to which they are applied. This thesis investigates a number of aspects of the application of the relatively new Ant Colony Optimisation (ACO) metaheuristic to different COPs.

The standard ACO metaheuristic is a constructive algorithm loosely based on the foraging behaviour of ant colonies, which are able to find the shortest path to a food source by indirect communication through pheromones. ACO's artificial pheromone represents a model of the solution components that its artificial ants use to construct solutions. Developing an appropriate *pheromone representation* is a key aspect of the application of ACO to a problem. Since its inception in the early 1990s, ACO has been applied to an increasing range of problems, leading to the development of a range of pheromone representations (Dorigo and Stützle, 2002). However, pheromone representations have typically been chosen in an *ad hoc* fashion, either by selecting a representation used previously for a similar problem or by developing a new representation that appears intuitively to suit the problem in question. The absence of a systematic approach to adapting ACO to different COPs has resulted in inconsistent performance across different problems.

An examination of existing ACO applications and the constructive approach more generally reveals how the metaheuristic can be applied more systematically across a range of COPs. The two main issues addressed in this thesis are biases inherent in the constructive process and the systematic selection of pheromone representations.

All constructive metaheuristics explore a tree of constructive decisions (a construction tree) the shape of which is determined by problem constraints and the chosen method of building solutions. The shape of this tree, combined with the mapping from paths in the tree to solutions, can bias a constructive search. This is particularly a problem in COPs where infeasible solutions cannot be avoided—such infeasible solutions have an elevated probability of being discovered by a constructive algorithm. This situation is common in COPs in which a number of entities (e.g., tasks, projects, or individuals) compete for limited resources. Alternative techniques for determining the order in which demands for

resources are considered can improve the probability of reaching feasible solutions in these types of COP. A number of these techniques are investigated, revealing that commonly used static and dynamic heuristics for this task generally work well, although cannot guarantee a good probability of finding feasible solutions across all problem instances.

The effectiveness of a given pheromone representation is related to how it maps onto the construction tree for a problem. Considering this finding, a systematic approach is developed for the selection of an appropriate pheromone representation based on characteristics of the COP being solved. The comparative performance of ACO using alternative pheromone representations—those used in the literature, suggested by the new system, or potentially applicable—is investigated for six problem types, including the travelling salesman, multiple knapsack, quadratic assignment, generalised assignment, shop scheduling and car sequencing problems. Results confirm that the algorithm's suggested pheromone representations generally perform well, with two main exceptions. First, an intuitively inappropriate pheromone representation previously unused with the knapsack problem was found to perform best on that problem, a finding that will be the subject of further investigation. Second, results for the car sequencing problem suggest that using the simplest pheromone representation that models solution identity (rather than structure) may be better than attempting to model the full complexity of solution characteristics that contribute to solution cost.

The systematisation of ACO should lead to more consistently high performance of the algorithm across different problems. Additionally, it supports the creation of a generalised ACO system, capable of adapting itself to suit many different combinatorial problems without the need for manual intervention.

Acknowledgements

No work this large could be produced without the help, hindrance or influence of a large number of people and organisations. A sample of them get a mention here.

Professionally, I would like to extend my thanks to

Marcus Randall, for his guidance, ideas, encouragement, prodding, persuading, understanding, assistance during tough times and even for his confident, if unsettling, assertion in recent times, "Now, you are the expert." I'm glad to call you my friend as well as my supervisor.

Tim Hendtlass, whose experience in supervision and many suggestions regarding the process of producing a thesis have helped immensely. Thanks also to his cohorts at the Centre for Intelligent Systems and Complex Processes at Swinburne University.

School of Information Technology, Bond University, for providing me with a scholarship to undertake my doctorate, for offering me the chance to teach some of my favourite undergraduate subjects, for the many friendly and helpful support staff, and for providing the most cohesive (yes, it is true) group of colleagues in the university.

The Australian Government, for providing me with an Australian Postgraduate Award during my PhD candidature.

On a personal level, I would like to thank

My officemates, past and present: *Dr Sun Zhaohao*, for his boundless enthusiasm and efforts to teach me Chinese; *Kieth Hales*, for putting up with the antics of us young folk; *Henry Larkin*, for having a sense of humour so strange it could be my own; and the younger *James Larkin*, for encouraging my juggling skills at every opportunity.

Bond Ultimate Frisbee, for providing many energetic and fun times and some of the best friends I could ever hope to meet, in particular business PhD candidate *Lisa Watson* and my IT colleague *Dr Phil Stocks*.

Friends too numerable to mention, for all the good times that helped balance my life and even for some of the bad (because life's like that).

Drs Bob Montgomery and Laurel Morris (Dad and Mum), for always having my best interests at heart and for frequently reminding me that there's more to life than working on one's thesis. The travel we've shared has been equally rewarding as my research. In particular, it was with their encouragement that early in my candidature I began my amateur underwater photography career, something which has brought me more joy than a whole bag of doctorates could ever do. And to all the unwilling aquatic victims of my camera: *So long, and thanks for all the snaps.*

Contents

1 Introduction **1**
- 1.1 Combinatorial Optimisation . . . 1
- 1.2 Evolutionary Computing and Collective Intelligence . . . 3
- 1.3 Optimisation Inspired by Ant Colony Behaviour . . . 5
- 1.4 Scope and Aim . . . 8
- 1.5 Thesis Outline . . . 8

2 ACO Algorithms and Applications **10**
- 2.1 Constructive Heuristics and Metaheuristics . . . 10
  - 2.1.1 Sequence Space and Solution Space . . . 12
  - 2.1.2 Randomness and Greed in Constructive Metaheuristics . . . 15
- 2.2 The ACO Metaheuristic . . . 16
- 2.3 ACO Algorithms . . . 18
  - 2.3.1 Ant System . . . 18
  - 2.3.2 Ant-Q and Ant Colony System . . . 20
  - 2.3.3 $\mathcal{MAX} - \mathcal{MIN}$ Ant System . . . 21
  - 2.3.4 Rank-based Ant System . . . 22
  - 2.3.5 Hyper-cube Framework for ACO . . . 22
  - 2.3.6 Other Variants . . . 23
- 2.4 ACO Applications . . . 24
  - 2.4.1 Subset Problems . . . 24
  - 2.4.2 Permutation/Routing Problems . . . 27
  - 2.4.3 Scheduling Problems . . . 29
  - 2.4.4 Assignment Type Problems . . . 33
  - 2.4.5 Constraint Satisfaction Problems . . . 36
  - 2.4.6 Other Problems . . . 37
- 2.5 Formalisations of ACO . . . 41
  - 2.5.1 Graph-based Ant System . . . 41
  - 2.5.2 Ant Programming . . . 42
- 2.6 Divergence of Construction Graph and Pheromone Representation . . . 43
  - 2.6.1 Constructive Heuristics and Virtual Construction Graphs . . . 45
- 2.7 Local Search in ACO . . . 46
- 2.8 Summary . . . 47

3 Bias in Constructive Heuristics **49**
3.1 Bias Inherent in the Constructive Process . . . 50
3.2 Variable Length and Infeasible Solutions . . . 52
3.3 Assignment Problems and Assignment Order . . . 55
3.3.1 Using Assignment Order to Reduce Infeasible Space . . . 58
3.4 Examples of Bias in Constructive Algorithms . . . 67
3.4.1 Solving the JSP, GSP or OSP as Assignment Problems . . . 69
3.4.2 Solving Assignment Problems with No Assignment Order . . . 70
3.5 Bias Effects and Problem Size . . . 71
3.6 Deliberately Introduced Sources of Bias . . . 72
3.7 Summary . . . 74

4 Pheromone Representations **76**
4.1 Formalisation of Pheromone Representations . . . 76
4.1.1 Representation-Oriented and Identity-Oriented Pheromones . . . 80
4.1.2 Atypical Pheromone Representations . . . 81
4.2 Using Higher Order Pheromone Representations . . . 84
4.2.1 Examples . . . 86
4.2.2 Notes on Higher Order Pheromone Representation Usage . . . 87
4.3 Summary . . . 90

5 Interaction Between Constructed Solution Biases and Pheromone **92**
5.1 Mechanisms of Interaction . . . 92
5.1.1 Low Level Interactions . . . 93
5.1.2 High Level Interactions and Competition-Balanced Systems . . . 95
5.2 Case Study: Pheromone Representations for the JSP, GSP and OSP . . . 100
5.3 Pheromone and Bias Interactions in Other Problems . . . 108
5.3.1 MKP . . . 109
5.3.2 GAP . . . 110
5.3.3 TSP and QAP . . . 110
5.4 Summary . . . 113

6 Selecting Pheromone Representations Considering Bias **115**
6.1 Controlling Bias Using the Pheromone Update . . . 116
6.2 Unique Representation in Pheromone . . . 117
6.2.1 Unique Representation and Bias . . . 119
6.2.2 Unique Representation in Higher Order Pheromones . . . 120
6.2.3 Examples of Unique and Multiple Representation . . . 120
6.2.4 Shared Representations in Pheromone . . . 123
6.3 Systematically Determining Appropriate Pheromone Representations . . . 125
6.3.1 Case Study: Car Sequencing Problem . . . 128
6.3.2 A Decision Algorithm for Pheromone Selection . . . 129
6.4 Potential to Counteract Bias . . . 133
6.5 Summary . . . 134

7 Pheromone Representation and Assignment Order Comparative Performance 136
7.1 Methodology . . . 136
7.1.1 Problems and Pheromone Representations . . . 137
7.1.2 Implementation of ACO Algorithms for Studied COPs . . . 138
7.1.3 Variables and Analysis Techniques Used . . . 140
7.2 Results . . . 142
7.2.1 TSP . . . 142
7.2.2 MKP . . . 143
7.2.3 GSP . . . 146
7.2.4 QAP . . . 147
7.2.5 GAP . . . 150
7.2.6 CSeqP . . . 155
7.3 Summary . . . 159

8 Conclusions and Further Work 163
8.1 Summary . . . 163
8.2 Contributions . . . 166
8.3 Conclusions and Future Work . . . 168

Bibliography 170

A Glossary of Problem Names and Acronyms 184

B Objective Functions Used in Pheromone Representation Selection 190

C Details of Non-benchmark Problem Instances Used 198

D Tables of Results 200

E Publications Arising from this Study 229

F Notation Used in Thesis 230

List of Tables

1.1 A continuum of collective intelligence approaches 5

2.1 Sample of ACO applications 25

3.1 Feasible probability under different assignment orders for `gap1` instances . 66
3.2 Feasible probability under different assignment orders for `gap2` instances . 66
3.3 Pairwise comparison of assignment orders for `gap1` instances 68
3.4 Pairwise comparison of assignment orders for `gap2` instances 68
3.5 Examples of bias in constructive algorithms 70
3.6 Bias effects and instance size 72

4.1 Common pheromone representations 80
4.2 Sample of ways to customise generic higher order pheromone equation . . . 88
4.3 Pheromone representations used in the literature and potentially applicable 91

6.1 Suggested pheromone representations for common COPs 135

7.1 Problems and pheromone representations studied 137
7.2 Parameter values used in ACO algorithm implementations 139
7.3 Values of β used with each problem 139
7.4 Heuristic information used by ACO and $\text{ACO}_{\text{undir}}$ algorithms 139
7.5 Example comparison table 141
7.6 TSP instances 142
7.7 TSP pheromone representation comparisons 143
7.8 MKP instances 144
7.9 MKP pheromone representation comparisons 145
7.10 GSP instances 146
7.11 JSP, GSP & OSP pheromone representation comparisons 147
7.12 OSP pheromone representation comparisons 147
7.13 QAP instances 148
7.14 QAP pheromone representation comparisons 149
7.15 GAP instances 150
7.16 GAP pheromone representation comparisons 152
7.17 GAP assignment order comparisons 154
7.18 CSeqP instances 156
7.19 CSeqP pheromone representation comparisons 158

D.1 TSP results, no local search 201
D.2 TSP results, with local search 202

D.3 MKP results 203
D.4 GSP results 206
D.5 QAP results 208
D.6 GAP results 212
D.7 CSeqP results 222

F.1 Notation used in thesis 230

List of Figures

2.1 Constructive hypercube 13
2.2 Construction tree for a small subset problem 14
2.3 Solution space for a small subset problem 15
2.4 Construction graph for a small subset problem 17
2.5 A small JSP instance 31
2.6 Alternative assignment type problem construction graphs 34
2.7 Constraint satisfaction problem alternative construction graph 37

3.1 A small JSP instance: `jsp2-2` 51
3.2 Construction tree for small example JSP instance 51
3.3 Solution probability for 15 and 50 item MKP instances 54
3.4 Construction trees for small GAP instance with different assignment orders 56
3.5 Distribution of feasible solution probabilities for `gap2` instances 59
3.6 Number of infeasible sequences against $P(\text{feasible})$ in `gap2-1` 60
3.7 Distribution of path lengths for first `gap2` instance under different assignment orders 63
3.8 Construction tree for `jsp2-2` using alternative construction approach . . . 71

5.1 Full details of `jsp2-2` instance 94
5.2 Construction tree for `jsp2-2` 94
5.3 Decisions affected by sequence $\langle 1, 2, 3, 4 \rangle$ in `jsp2-2` 96
5.4 Decisions affected by sequence $\langle 1, 3, 4, 2 \rangle$ in `jsp2-2` 97
5.5 Details of `jsp3-3` and `osp3-3` instances 101
5.6 Line scheduling factor against various measures for `jsp3-3` and `osp3-3` . . 103
5.7 Line scheduling factor against solution cost using ACO 105
5.8 Line scheduling factor against sequence probability using ACO 105
5.9 Solution characteristic usage frequency against corresponding solution cost 107
5.10 Potential bias interaction in the `mknap1-15item` MKP instance 111
5.11 Potential bias interaction in the `gap2-1` GAP instance 112

6.1 Use of TSP-like pheromone with a small subset problem 118
6.2 Small example OSP instance 123
6.3 Decision tree for selection of pheromone representations 131

Chapter 1

Introduction

1.1 Combinatorial Optimisation

Any problem in which solution quality may be objectively quantified and for which one seeks the best solution is an *optimisation* problem. Such problems typically involve a mathematical *model* of some aspect of the real world, with variables that are either continuous or discrete. The problems of interest in this work involve discrete variables and are known as *combinatorial optimisation problems* (COPs), as their solutions are represented by combinations of discrete components. Scheduling, design, layout, distribution (e.g., of mail and other commodities), timetabling, resource allocation and project management all feature problems of this kind. While such problems are found in industrial and governmental activities as well as in activities unrelated to work, it is typically only in the former two areas that it is worthwhile expending significant effort to find good solutions. Finding the best solution in these fields can lead to reductions in expenditure and increases in profit (not directly related to reduced expenditure) (Osman and Kelly, 1996). Environmental benefits may also result if environmental aspects of the problem are considered to be important and explicitly included in the problem model. Consequently, the study of efficient techniques to solve these problems is of great importance.

To objectively assess the relative merits of different solutions, the definition of a COP includes an *objective function*. The goal of combinatorial optimisation is to find a solution such that the value of the objective function is either maximised or minimised (depending on the problem). More formally, a COP may be defined as

$$Optimise\ f(x) \tag{1.1}$$

subject to (s.t.)

$$x \in \mathbb{X} \subset \Omega \tag{1.2}$$

where x is a solution to the COP being solved, $f(x)$ is the objective function, $\mathbb{X}$ is the set of feasible solutions and Ω is the set of all possible combinations of components involved in the problem (Osman and Kelly, 1996).

COPs are typically easy to describe and the value of the objective function can be calculated in polynomial time (Osman and Kelly, 1996). In the absence of specialised insight into the structure of a problem and its solutions, the only way to guarantee an optimal solution is to implicitly or explicitly enumerate each solution and evaluate its cost. However, as problem size grows, the number of combinations of components representing potential solutions typically grows exponentially, making complete exploration of the search space impracticable. Thus, the majority of COPs of interest are NP Complete (Garey and Johnson, 1979). That is, there exist no known polynomial time algorithms that can find the optimal solution to these problems.

Two broad approaches are available for solving COPs: exact algorithms and heuristics. Exact algorithms such as Branch and Bound, Branch and Cut and A* (Winston, 1992) use a variety of mathematical techniques to identify parts of the solution space that cannot contain the optimal solution and which consequently need not be explored.[1] Given sufficient computing time, such algorithms can guarantee an optimal solution. However, in the worst case these algorithms still suffer from exponential run times (Anderson, Sweeney and Williams, 1994).

Heuristics are polynomial time algorithms that can find good, but not necessarily optimal, solutions to COPs in a reasonable amount of computational time. Heuristics vary in their degree of problem-specificity. Specialised heuristics are tailored to suit only one problem or a small number of related problems. Such heuristics employ knowledge of the problem domain to produce high quality solutions, often very quickly. However, their applicability is then restricted to those problems. Metaheuristics operate at a more abstract level, directing a problem-specific heuristic through solution space. Each metaheuristic is based on one or more principles concerning *how* the search should be conducted, rather than focussing on the individual solution characteristics that are of interest to a specialised heuristic. Metaheuristics, in particular those inspired by naturally occurring "optimisation" processes, have been used to produce good solutions to many difficult problems in recent decades (Corne, Dorigo and Glover, 1999; Glover and Kochenberger, 2002).

[1] See Taha (1992), Winston (1992) or Winston (1991) for detailed treatments of each of these approaches.

1.2 Evolutionary Computing and Collective Intelligence

Complex structure and behaviour can arise naturally as a result of apparently simple rules (Cohen and Stewart, 1994; Stewart and Cohen, 1994). Many naturally occurring processes appear to be intelligent, although this "intelligence" does not connote consciousness or intent. For several decades, computer scientists have used many of these intelligent natural processes as metaphors for computer-based algorithms to solve various problems, including COPs.

A notable example of a natural process that has been emulated to solve human problems is the *evolution* of species, in particular evolution within species. The mechanisms that underpin this evolution, while numerous, are generally quite simple. Different optimisation approaches based on the metaphor of species evolution, collectively known as *Evolutionary Algorithms* (EAs), emphasise different subsets of these mechanisms. While there are numerous texts that give detailed treatments of the actual mechanisms involved in natural evolution (e.g., Cohen and Stewart, 1994; Darwin, 1859, reprinted in 1988; Dawkins, 1986), it is sufficient to describe just the central idea of Darwinian evolution as it is this that is common to all EAs and most other nature-inspired optimisation techniques.

The central tenet of Darwinian evolution is that individual organisms have heritable characteristics that affect the probability of those organisms producing offspring that will carry those characteristics into future generations. In the field of evolutionary computation (of which EAs are a prime example) this has been transformed into the notion that characteristics that appear in better solutions should be used more frequently in subsequently produced solutions than those characteristics that appear in poorer solutions.

A number of EAs for learning and optimisation have been developed. Most of these approaches maintain a population of individuals representing solutions to the problem at hand, which are modified and, in some cases, bred. In the 1960s, Fogel (1997, 1999) developed *Evolutionary Programming* (EP), in which a collection of finite state automata (FSAs) were evolved to be better able to make predictions. EP applications include predicting the primality of numbers, learning of game strategies and training artificial neural networks. At the same time, Rechenberg (1973) and later Schwefel (1975) developed *Evolution Strategies* (ES), an algorithm for the optimisation of functions with real-valued variables. In ES, a new solution is produced from an existing one by making random adjustments to the values of its variables, an operation analogous to the random mutations introduced into an organism's genes at each generation. Both EP and ES are characterised as working on an individual's *phenotype*, or observed characteristics (Fogel, 1995). In contrast, *Genetic Algorithms* (GAs), developed in the mid 1970s by Holland (1975), are characterised as

manipulating an individual's *genotype*, as solutions are typically encoded as fixed-length strings similar to the chromosomes that carry genes in living organisms.[2] GAs maintain a population of solutions, manipulating them by operators such as crossover/recombination (inspired by the distinct biological processes of crossover during gamete production and sexual reproduction) and mutation (inspired by the biological process of the same name). As GAs operate on an encoded form of the solution, they can be applied to any COP for which a suitable encoding can be developed. However, traditional GA approaches typically have poor constraint handling capabilities, as their recombination operator can easily produce encodings that do not correspond to feasible solutions. Consequently, traditional GAs have had mixed success in combinatorial optimisation (Michalewicz, 1996).

The mechanisms for evolving improved solutions over time need not be based on the rules underlying the evolution of the species. Evolutionary computation is therefore broader than just those algorithms modelled on this kind of evolution. Another natural process adapted and emulated for learning and optimisation is the cooperative behaviour of animals such as social insects and flocking birds. *Swarm Intelligence* is a learning and optimisation technique in which individuals are associated with and act to create potential solutions, rather than directly representing solutions as in EAs. The two most notable examples of swarm intelligence are Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 1995), based on the flocking or swarming behaviour of birds or insects respectively, and Ant Colony Optimisation (ACO), based on the way ants communicate indirectly through their environment about paths to food. Although neither of these approaches is based on the metaphor of the evolution of species, algorithms from both exhibit the evolution of improved solutions over time and are thus examples of evolutionary computation. Indeed, a number of similarities have been identified between ACO and the EA Population-Based Incremental Learning (Monmarché, Ramat, Dromel, Slimane and Venturini, 1999).

All of these approaches may also be viewed as examples of *collective intelligence*, where knowledge about a problem is not stored centrally but is distributed either across individuals or in the environment in which individuals search (Bonabeau, Dorigo and Theraulaz, 1999). EAs, PSO and ACO can been placed on a continuum based on the extent to which individuals represent solutions versus how much control individuals have over the solutions produced, as shown in Table 1.1. Thus, whether to label a technique as an example of evolutionary computation or collective intelligence is more a matter of degree rather than of hard categorisation. In EAs, the population of individuals are solutions, and are acted on by an external environment. The current population therefore represents all the knowledge the algorithm has developed concerning the problem. Thus, unless explicitly incorporated into a particular EA, such algorithms have no memory of past solutions or

[2]While Holland (1975) emphasised the importance of encoding solutions as binary strings, modern GAs typically use encodings that are designed to suit the problem being solved (Michalewicz, 1996).

Table 1.1: A continuum of collective intelligence approaches.

	Approach		
	EAs	PSO	ACO
Knowledge of solutions	Individuals	Individuals' positions in solution space	Environment
Manipulation of solutions	Environment acting on individuals	Individuals acting on selves	Individuals acting on environment

the solutions encountered to reach the current population. In PSO, individuals know their location in solution space (and so indirectly represent solutions) and act on themselves and each other to move to new solutions. Individuals typically have limited historical information, as their respective vectors in solution space may be retraced by a limited number of steps. In ACO, individuals serve solely to construct solutions influenced by their environment, which in turn is modified by the solutions produced. Their environment therefore serves to remember solutions created in the past and their quality. Emulation of the real-world process of attenuation in the chemicals ants use to communicate means that the memory, and hence influence, of old solutions gradually fades.

1.3 Optimisation Inspired by Ant Colony Behaviour

Social insects such as termites, ants, bees and some species of wasps are capable of complex and intelligent group behaviour (Bonabeau et al., 1999). This collective behaviour is of interest as it emerges from apparently simple, and often indirect, interactions between members of these insect groups, known as *colonies*. An early attempt to understand these phenomena is due to Grassé (1959, cited in Bonabeau et al., 1999), who studied column formation inside termite nests. Grassé observed that termites, in a nest initially devoid of internal structure, can cooperate to create galleries within the nest without direct communication or coordination. One of the individual behaviours underlying this emergent group behaviour is that a single termite, having collected a soil particle from somewhere in the nest, is more likely to deposit the particle near to existing piles of dirt than elsewhere. Over time, positive feedback effects lead to the formation of well defined columns, which then join to form arches. Ultimately the space between the columns and arches is filled to form walls. The term *stigmergy* is used to describe this kind of behavioural feedback (Grassé, 1959). Stigmergic effects are present in most collective behaviour of social insects. Many of these interactions are mediated by *pheromones*, volatile chemicals secreted by individual insects that cause changes in other individuals' behaviours.

Ants use pheromone for communication in a number of contexts, including alerting other individuals to a threat, attracting mates, recognising individuals from the same colony, recognising a colony's gyne (i.e., queen), marking a colony's home range, marking trails to food sources and recruiting other ants to collect food from those sources (Vander Meer, Breed, Winston and Espelie, 1997). The way in which many ant colonies forage for food is of particular interest as a metaphor for combinatorial optimisation.

When an ant encounters a food source it retraces its path back to the nest and deposits a trail pheromone on the ground. Trail pheromone is often composed of two separate pheromones, an enduring pheromone to mark the trail and a more volatile pheromone used to recruit other ants to follow the path to the food source (Vander Meer et al., 1997). However, it is convenient when modelling the process to treat the two pheromones as a single pheromone serving both purposes. Observing that colonies of many species of ant can identify the shortest path to a food source using primarily pheromone-mediated communication, Deneubourg, Aron, Goss and Pasteels (1990, cited in Bonabeau et al., 1999) conducted a series of experiments with real ants in an artificial environment. A number of laboratory ants had their nest placed at one end of a fairly simple maze, where two paths of different lengths led to the same food source. Ants would choose between the two paths probabilistically, biased by the intensity of the pheromone trail on each. Initially, when pheromone trails were non-existent or weak, ants chose each of the two paths with uniform probability. However, due to the so-called *differential path effect*, a greater number of ants were able to traverse the shorter path to and from the food source than the longer path over the same period of time, leading to relatively more pheromone being deposited on the shorter path. This produced an *autocatalytic*[3] effect where a greater proportion of ants were attracted to the shorter path because of its higher pheromone levels, producing an even more rapid increase in those levels. Over time, pheromone on the longer path would evaporate and, in the absence of reinforcement, dwindle to negligible levels, so that eventually all ants chose the shorter path to and from the food source.

There are effectively a finite number of paths that ants may take to a food source, each of which has a different length. Finding the shortest path is therefore an optimisation process. Dorigo, Maniezzo and Colorni (1991) identified this feature of ant behaviour as a possible metaphor for a computer-based optimisation algorithm and produced the first ACO algorithm by adapting existing models of trail pheromone dynamics to suit combinatorial optimisation.[4] Their initial applications focused on the well-known travelling salesman problem (TSP), as there is a correspondence between ants finding the shortest path to a food source and identifying the shortest Hamiltonian cycle in a weighted graph. Dorigo, Di Caro and Gambardella (1999) developed a formal definition of ACO, bring-

[3] An autocatalytic process is one in which the catalyst is one of the products of that process.

[4] The seminal work in this area is Dorigo's (1992) PhD thesis.

ing a number of similar ant-inspired optimisation algorithms into the one metaheuristic framework.

While the majority of metaheuristics are *iterative* (i.e., they iteratively alter a complete solution or set of solutions), ACO algorithms are typically *constructive*. Just as a real ant makes a series of decisions about which subpath to take from each point along its overall path, ACO algorithms use a collection of artificial ants to build complete solutions from a predefined set of solution components. The next chapter provides a technical definition of constructive metaheuristics and the ACO metaheuristic and its applications.

As ACO is a learning algorithm it shares some similarities with other learning techniques such as *artificial neural networks* (see, e.g., Picton, 1994) and *reinforcement learning* (see, e.g., Sutton and Barto, 1998). All three techniques use forms of reinforcement to make certain behaviours or outputs more likely. For instance, an artificial neural network, modelled on the operation of neurons in animal brains, learns by altering the strength of relationships between the artificial neurons that make up the network in order to improve the accuracy of its output. This contrasts with ACO where learning occurs by modifications to the environment in which artificial ants search. Reinforcement learning is a broad field that can be said to contain ACO, as in the most general sense reinforcement learning involves an *agent* (e.g., an artificial ant) responding to its environment to take an action it has learned is good (e.g., selecting a particular solution component based on current pheromone levels). The relationship between the two are described in more detail by Nowé and Verbeeck (1999).

ACO has rapidly become a popular optimisation technique, applied to a diverse range of COPs. These include benchmark problems such as the TSP, constraint satisfaction problems, quadratic and generalised assignment problems, as well as more real-world problems such as machine scheduling and routing in packet-switched networks. There are two main reasons for the rapid adoption of ACO by researchers in the field of optimisation. First, nature-inspired optimisation algorithms are appealing because they can often be quite powerful despite the simplicity of the low level rules that govern them. Even in the absence of satisfying explanations to bridge the low level behaviour of the algorithm with the emergent optimisation behaviour, emulation of the real life low level behaviour of ants has led to the development of a good algorithm. Despite this, ACO has demonstrated its effectiveness on many of the problems to which it has been applied (Dorigo and Stützle, 2004).

Bonabeau et al. (1999) claim that the ACO approach may be particularly effective in dynamic optimisation problems, where the problem specification changes over time, necessitating changes to the solution. While the majority of ACO applications to date have been to static problems, those that have dealt with dynamic problems have often performed well (Dorigo and Stützle, 2004).

1.4 Scope and Aim

Ideally, a metaheuristic may be applied to an optimisation problem with no modification except for indicating the nature of solution representations over which the search is to take place. This is the aim of the emerging field of hyper-heuristics (Burke, Hart, Kendall, Newall, Ross and Schulenburg, 2002). However, in practice, metaheuristics are typically modified much more extensively to allow them to compete with specialised heuristics. Adapting a metaheuristic to suit a particular problem can be a time-consuming and non-systematic process. Recent work by Randall (1999) and Randall and Abramson (2001) has shown that it is possible to produce a generalised iterative metaheuristic that does not require extensive modification to be applied to different COPs. This generalised system has had a great deal of success on a wide range of problems. An obvious conclusion from these works is that the adaptation of iterative metaheuristics can be largely automated. However, constructive metaheuristics, such as ACO and Greedy Randomised Adaptive Search Procedures (GRASPs), are less amenable to generalisation (Randall, 2002a). From this it can be inferred that it is more difficult to *systematise* their adaptation. Indeed, ACO is typically adapted to suit each new problem in an *ad hoc* fashion, with the resulting algorithms becoming increasingly dissimilar to the biological metaphor on which the metaheuristic is based. This has produced a mixture of successes and failures in its application.

This work investigates the nature of constructive metaheuristics and particular aspects of the ACO metaheuristic that affect how they should be applied to different COPs. The findings may be used to apply ACO more systematically and hence with greater confidence that its performance will be good. In those problem domains to which ACO has previously been applied, some alternative approaches may be suggested. The findings should also support the application of ACO to problems it has not previously been used to solve.

1.5 Thesis Outline

Chapter 2 provides a formal introduction to constructive metaheuristics in general and the ACO metaheuristic in particular. It also gives an overview of the numerous applications of the ACO metaheuristic to combinatorial optimisation problems. The established approach to adapting ACO to suit a given COP is to model the problem in terms of a graph which ants traverse to create solutions. The advantages and disadvantages of this approach are also discussed in Chapter 2.

Chapter 3 explores search biases inherent in any constructive optimisation algorithm and how these biases depend on the COP being solved and the constructive algorithm applied. Chapter 4 presents a formal treatment of the definition and use of pheromone,

while Chapter 5 discusses the interaction between constructive heuristic biases and different pheromone representations.

Based on the examination of existing uses of pheromone in Chapter 2, and the investigation of constructive heuristic bias and its potential impact on ACO's performance described in Chapters 3 and 5, Chapter 6 presents a more systematic approach to the selection of pheromone representations.

Chapter 7 presents the results of empirical investigations of the relative performance of alternative pheromone representations for different problems. Chapter 8 summarises the contributions of the overall investigation and describes future research directions.

Chapter 2

ACO Algorithms and Applications

This chapter provides a technical introduction to constructive algorithms for combinatorial problems and a more detailed definition of the ACO metaheuristic. The development of ACO is presented together with a number of key ACO algorithms. Section 2.4 reviews many of the applications of ACO to different COPs. This review illustrates both the diversity of ACO's application and common ways in which it has been adapted. Two key formalisations of ACO are described in Section 2.5, while a number of observations concerning the application of ACO to different problems are made in Section 2.6.

2.1 Constructive Heuristics and Metaheuristics

In a broad sense there are two kinds of heuristics, iterative and constructive. Iterative heuristics perform successive modifications to one or more complete solutions, transforming them into new solutions. The majority of optimisation algorithms are of this kind, including local search and metaheuristics such as Simulated Annealing (SA) (van Laarhoven and Aarts, 1987), Tabu Search (TS) (Glover and Laguna, 1997), most EAs and PSO. In contrast, constructive heuristics start with an initially empty solution and add to it in order to produce a complete solution.

Constructive heuristics take an empty solution $s = \varnothing$ and successively add *solution components* to build a complete, typically feasible, solution to a problem. Denote the set of solution components by $\mathfrak{C}$, and a single solution component by $\mathfrak{c}_i \in \mathfrak{C}$. As constructive heuristics typically add solution components one at a time, a solution produced by such heuristics may be represented as a sequence of solution components $\mathfrak{s} = \langle \mathfrak{c}_i, \ldots, \mathfrak{c}_k \rangle$. Partial sequences are represented by $\mathfrak{s}^p$, while a set of sequences is denoted by $\mathfrak{S}$. The i^{th} component in $\mathfrak{s}$ is denoted by $\mathfrak{s}[i]$. At each step of a constructive heuristic, the set of solution components that may be added to the partial solution $\mathfrak{s}^p$ is given by $\mathfrak{N}(\mathfrak{s}^p) \subseteq \mathfrak{C}$. Thus $\mathfrak{N}(\mathfrak{s}^p)$ is similar to the *neighbourhood* of a solution in an iterative heuristic.

Algorithm 1 Generic constructive (meta)heuristic

1: **while** stopping criterion not met **do**
2: $\quad\mathfrak{s} \leftarrow \langle\rangle$
3: $\quad$**while** $\mathfrak{s}$ is incomplete **do**
4: $\quad\quad$Select a component $\mathfrak{c} \in \mathfrak{N}(\mathfrak{s})$
5: $\quad\quad\mathfrak{s} \leftarrow \mathfrak{s} + \mathfrak{c}$
6: $\quad$**end while**
7: $\quad$**if** first iteration **or** $\mathfrak{s}$ is better than $\mathfrak{s}^{best}$ **then**
8: $\quad\quad\mathfrak{s}^{best} \leftarrow \mathfrak{s}$
9: $\quad$**end if**
10: **end while**
11: Output $s_{\mathfrak{s}^{best}}$

The constructive algorithm used to build solutions, denoted by $\mathcal{A}$, defines both the components in $\mathfrak{C}$ and the set of available components $\mathfrak{N}$. Hence, $\mathcal{A}$ is assumed to be tailored to suit one COP. $\mathcal{A}$ also defines a mapping from sequences to solutions, where $s_{\mathfrak{s}}$ is used to denote the solution represented by the sequence $\mathfrak{s}$. Therefore, implicitly, $\mathcal{A}$ defines the set of solutions which each partial sequence $\mathfrak{s}^p$ may become, denoted by $S_{\mathfrak{s}^p}$. In many constructive algorithms, there is a many-to-one relationship between sequences and solutions. Hence, the set of sequences that represent a solution s is denoted by $\mathfrak{S}_s$.

An outline of a constructive heuristic in which multiple solutions are constructed over a number of iterations is presented in Algorithm 1. As the distinction between a heuristic and a metaheuristic is one of problem specificity, the algorithm shown could actually be either. However, metaheuristics are typically characterised by the creation of multiple solutions during a run, while problem-specific heuristics more commonly generate only a single solution. A deterministic heuristic may be described by lines 2 through 6 of Algorithm 1, with the addition that the single solution produced is output.

The nature of the solution components depends on the problem specification (as well as choices of the constructive heuristic's designer). For instance, in the travelling salesman problem (TSP), where solutions may be represented as permutations of the cities that must be visited, $\mathfrak{C}$ may be defined to represent the set of cities. A sequence that is a permutation of the elements of $\mathfrak{C}$ will then correspond to a valid solution.

The formulation of a COP involves a number of *decision variables*, each with its own *domain* of acceptable values. Each distinct solution corresponds to a different set of values assigned to these variables. For instance, in the TSP of n cities, n decision variables may be defined, one for each city, each representing the next city to be visited. In a subset problem, where a set of items must be chosen from some larger set subject to various constraints, the decision variables may represent the presence or absence of a particular item in the chosen subset. Typically, the addition of a solution component corresponds

to the assignment of a value to one of the decision variables in the problem, although a single solution component might not on its own represent the binding of a value to a variable. For instance, in the TSP, adding component $\mathfrak{c}_k$ to the sequence $\langle \mathfrak{c}_i, \ldots, \mathfrak{c}_j \rangle$ equates to assigning the value $\mathfrak{c}_k$ to the decision variable for city $\mathfrak{c}_j$, i.e., edge $(\mathfrak{c}_j, \mathfrak{c}_k)$ is now part of the solution being constructed. Blum (2004) refers to this kind of solution component as a *natural solution component*, because sequences are most naturally built from these kinds of components even though an individual component has no meaning separate from its place in a sequence.

Solution components may also more directly represent the binding of a value to one of the decision variables in a problem. For instance, it is commonplace in constructive algorithms for subset problems that the set of solution components $\mathfrak{C}$ corresponds to the set of items from which a subset must be chosen. Hence, adding solution component $\mathfrak{c}_i \in \mathfrak{C}$ is equivalent to setting the corresponding item's decision variable to true. Section 2.4 gives a number of examples of problems and the way that solution components have been defined for them.

As the addition of a solution component decides the value of one of the variables in the problem, it consequently restricts the set of solutions that the partial solution may become.[1] This is illustrated in Figure 2.1, which shows the solution space for a problem with three binary variables.

2.1.1 Sequence Space and Solution Space

The spaces of sequences and corresponding solutions that constructive algorithms explore can be viewed in a number of ways. Recently, a number of authors have recognised that the *sequence space* that constructive algorithms explore forms a tree of constructive decisions (Birattari, Di Caro and Dorigo, 2002; Maniezzo, 1999; Maniezzo and Milandri, 2002; Montgomery, Randall and Hendtlass, 2004). There is no established name for such a tree, so the term *construction tree* will be used hereafter. This is similar to the tree explored by Constraint Logic Programming (CLP) algorithms (Jaffar and Maher, 1994), where each level in the tree corresponds to one of the decision variables in the problem and nodes at the same level correspond to alternative values for that decision variable. However, whereas CLP algorithms explore (explicitly and implicitly) this tree in a systematic fashion, constructive algorithms explore a diverse, but not exhaustive, range of branches by successively constructing solutions. Furthermore, the order in which decision variables are assigned values may vary between paths in a construction tree, while in the tree explored

[1] If solutions are constructed using a more biologically inspired approach, where decisions made later in the constructive process can effectively undo previous decisions, much as some genes can switch other genes on or off, then this would not be the case. However, it is more typical that solutions are constructed in a straightforward manner, with the values of independent decision variables being decided at each step.

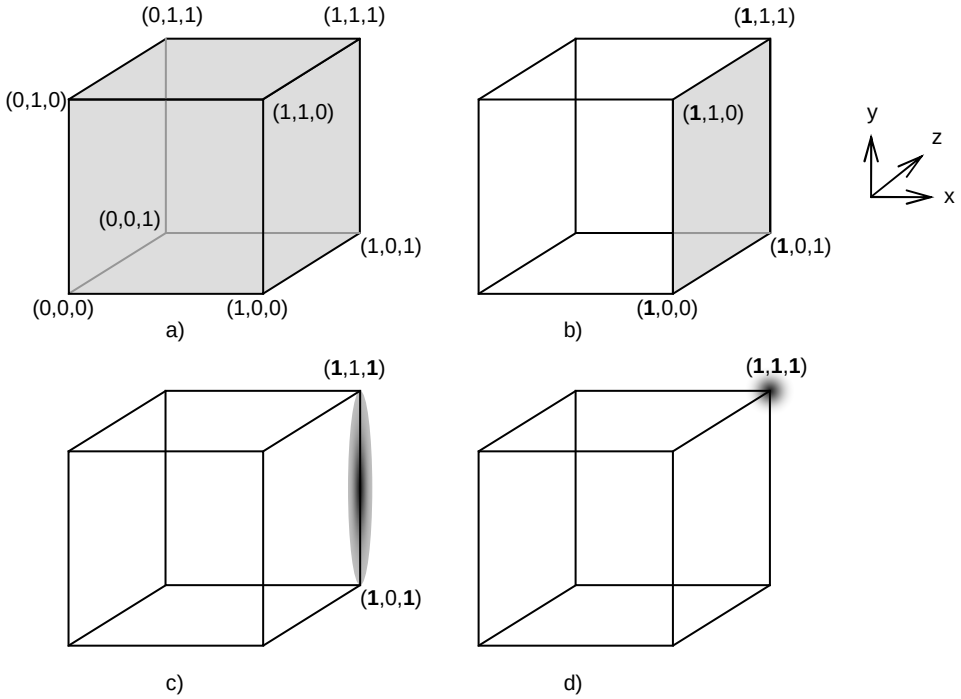


Figure 2.1: Solution space for combinatorial problem with three binary decision variables; all solutions are feasible. The shaded area shows the constructive neighbourhood of the partial solution at each step. The four cubes show the neighbourhood after (a) no variables have been assigned, (b) variable x has been assigned, (c) variable z has been assigned, (d) after variable y has been assigned.

by a CLP algorithm this order is fixed.

The construction tree defined by a constructive algorithm $\mathcal{A}$ is denoted by $\mathcal{T}_{\mathcal{A}}$. The root of $\mathcal{T}_{\mathcal{A}}$ corresponds to the empty sequence $\langle\rangle$ and consequently to the solution $\varnothing$, while leaves correspond to completed (or infeasible partial) sequences, and so to completed or infeasible partial solutions. Hence, a partial sequence $\mathfrak{s}^p$ corresponds to an incomplete path in the tree, and so to a partially completed solution s^p. Although trees are by definition undirected, the majority of constructive heuristics do not allow backtracking (i.e., they cannot remove solution components and thereby retrace their steps) and hence edges in the tree are directed away from the start node $\langle\rangle$.

An example construction tree for a simple subset problem is given in Figure 2.2. The subset problem in question consists of selecting any three items from a set of four. When depicting a construction tree, either edges or nodes may be labelled with the component chosen at each step. Note that in a mathematical definition of such a tree each node would be identified by the partial sequence it represents. However, to simplify the presentation of the tree each node is only labelled with the solution component chosen to reach that point in the tree, while the partial sequence is read by tracing a subpath starting at the start node $\langle\rangle$.

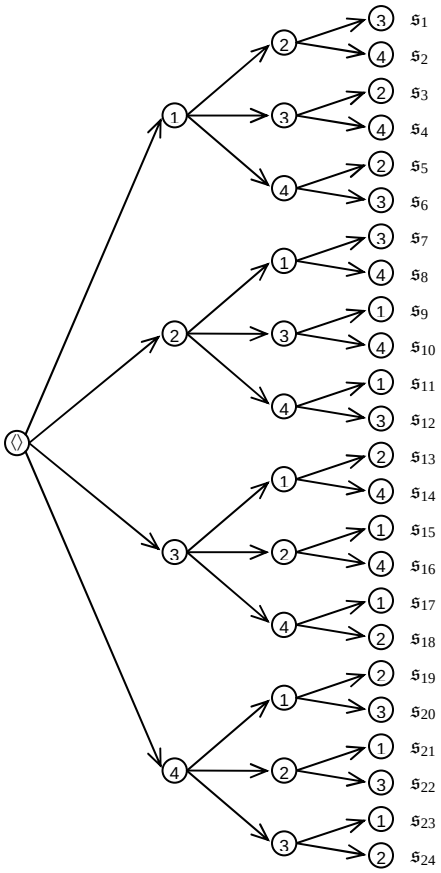


Figure 2.2: Construction tree for a simple four item subset problem in which three of the four items must be chosen. The set of components $\mathfrak{C} = \{1, 2, 3, 4\}$.

An alternative way of representing the solution space that constructive algorithms explore is as a graph showing the partial solutions represented at each step. In some problems this collapses a number of nodes in the corresponding construction tree into one. The partial solution space (i.e., space of partial solutions) for the same four item subset problem is shown in Figure 2.3.

These alternative ways of representing the spaces that constructive algorithms explore offer different perspectives on the operation of such algorithms. Study of the sequence space defined by the construction tree in particular forms the basis of the next chapter. Furthermore, they offer an alternative to the traditional way in which the constructive process is described in ACO.

Table F.1 in Appendix F summarises the notation used in the thesis to describe constructive heuristics. This page is duplicated as a fold-out page so that it may be kept open while reading other parts of the thesis.

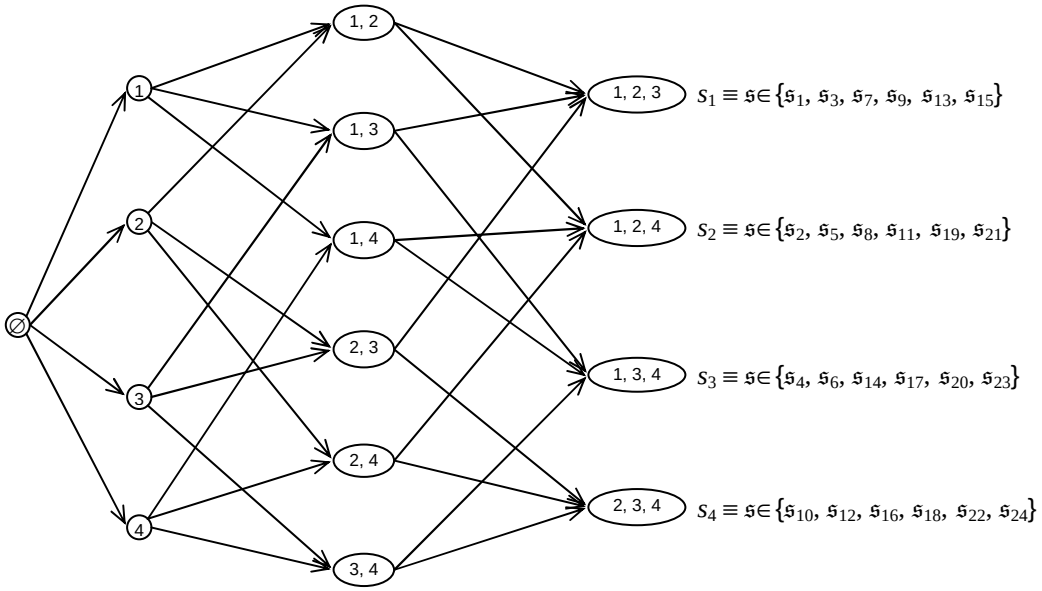


Figure 2.3: Constructed solution space for a simple four item subset problem in which three of the four items must be chosen. The set of components $\mathfrak{C} = \{1, 2, 3, 4\}$.

2.1.2 Randomness and Greed in Constructive Metaheuristics

The simplest constructive algorithm selects successive components randomly until a solution is built (or until no feasible solution is possible). This corresponds to replacing line 4 of Algorithm 1 with **Select a component from $\mathfrak{N}(\mathfrak{s})$ randomly**. In general, random construction produces poor solutions and improved results may be obtained if component selection is biased according to some (typically short-sighted) heuristic measure of each component's utility (Dorigo and Stützle, 2002).[2] The extreme form of this approach is a simple greedy constructive algorithm which selects, at each step, the component with the best heuristically estimated utility. Such an algorithm is consequently deterministic and so capable of producing only a single solution. While greedy algorithms in general should intuitively produce better solutions than randomly sampling the search space, constructing solutions from components that individually appear good does not necessarily result in a good solution. For instance, the *nearest neighbour* heuristic is a constructive algorithm that may be applied to the TSP. It begins by selecting a city at random and then successively selects the nearest unvisited city as its next destination until a complete solution (a permutation of the cities) is produced. The solutions created by this heuristic are often characterised by relatively large distances between cities that were added towards the end of the constructive process, as purely greedy choices made earlier leave no good alternatives later in construction.

[2]Nevertheless, an otherwise undirected, randomised constructive algorithm provides an important benchmark against which to measure the performance of more specialised algorithms. Such an algorithm is an important part of analyses in Chapters 3, 5 and 7.

More complex constructive metaheuristics lie between these two extremes, combining some mechanism for exploring a range of solutions with a bias towards components that appear to be good. For example, Greedy Randomised Adaptive Search Procedures (GRASPs) (Feo and Resende, 1995) are a class of constructive metaheuristics in which, at each step, a component is chosen randomly from a small set of components selected according to some greedy rule. Another key example is the ACO metaheuristic, in which components are (usually) selected probabilistically according to a distribution biased by a (typically greedy) heuristic measure of component utility adjusted by an estimate of component utility learned over time.

2.2 The ACO Metaheuristic

The first ACO algorithm, Ant System (AS), is due to Dorigo et al. (1991), and is more fully described by Dorigo (1992). Several variations of Ant System were later developed, eventually leading to the formalisation of the approach as Ant Colony Optimisation by Dorigo and Di Caro (1999) and Dorigo, Bonabeau and Theraulaz (2000). This section describes the ACO metaheuristic, while the next section describes some notable variations of the original Ant System upon which ACO is largely based.[3]

An ACO algorithm consists of a number of iterations of solution construction, within which a number of simple agents called (artificial) ants each construct a solution. Hence, each ant's behaviour follows Algorithm 1 between lines 2 and 6.

According to the early ACO literature (Dorigo et al., 2000), an ACO algorithm (i.e., an implementation or instance of the ACO metaheuristic) is intended to find minimum costs paths over a graph $G = (\mathfrak{C}, L, W)$ while respecting a set of constraints Ω. In this formulation $\mathfrak{C}$ is a set of solution components (as defined in the previous section), $L = \{l_{\mathfrak{c}_i\mathfrak{c}_j} | \mathfrak{c}_i, \mathfrak{c}_j \in \mathfrak{C}\}$ is a finite set of possible *connections* between the elements of $\mathfrak{C}$, and W a set of weights associated with the components $\mathfrak{C}$ or the connections L or both. Hence, for the TSP, $\mathfrak{C}$ is the set of cities, L is the set of edges connecting cities and W is the weight of those edges. Ω is defined as $\Omega(\mathfrak{C}, L, t)$, a finite set of constraints assigned over the elements of $\mathfrak{C}$ and L at a given time t (although for many problems the set of constraints will not change with time). This definition is closely related to descriptions of actual ant behaviour and the artificial environment experiments of Deneubourg, Aron, Goss and Pasteels (1990). While it is well-suited to many of the problems to which ACO was applied up to 1999, it is not as general as the actual application of the metaheuristic has been. Accordingly, current definitions of ACO are not presented in terms of ants solving shortest-path problems.

[3]The notation in this section has in places been changed from the original sources to match that introduced in the previous section.

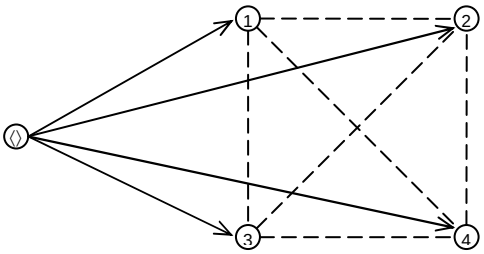


Figure 2.4: Construction graph for a simple four item subset problem in which three of the four items must be chosen. The set of components $\mathfrak{C} = \{1, 2, 3, 4\}$. All ants must start at node $\langle\rangle$ and move towards a component. Dashed lines represent undirected edges that are implicitly assigned a direction as they are traversed by an ant.

According to more recent descriptions, such as Dorigo and Stützle (2004), ants construct solutions by walking in a *construction graph* $G_C = (\mathfrak{C}, L)$, where L fully connects the elements of $\mathfrak{C}$. Hence, the construction graph defines what sequences may be constructed. Rather than finding a minimum cost path in this graph, newer descriptions state the goal of ACO as finding an optimal solution. A possible construction graph for the simple subset problem described earlier is presented in Figure 2.4. To ensure that ants do not make choices that produce infeasible sequences (and hence, solutions), extra constraints typically must be defined. Hence, the construction graph on its own is insufficient to describe the constructive process. These constraints, together with the construction graph, implicitly define the construction tree and space of partial solutions. That is, the set of all feasible paths in the construction graph form the paths of the construction tree.

ACO belongs to the class of model-based search (MBS) algorithms (Zlochin and Dorigo, 2002). In an MBS algorithm, new solutions are generated using a parameterised probabilistic model, the parameters of which are updated using previously generated solutions so as to focus the search on promising areas of the solution space. In ACO, the model is an analogue of the trail pheromones used by real ants, known in ACO as a *pheromone representation*. Early definitions of ACO specified that this artificial pheromone (referred to as pheromone hereafter) can be associated with the connections L, components $\mathfrak{C}$, or some combination of both, although a diverse range of pheromone representations have been developed since ACO's inception that do neither of these. An individual pheromone value is denoted by τ, and is typically a positive real-valued number. Hence, in the majority of ACO implementations, the set of pheromone values may be denoted by $T \subset \mathbb{R}^+$. As with real ants, pheromone is used to learn about good paths to take, and so possibly indirectly about good solutions to produce. Hence, pheromone information influences decisions concerning component selection. In addition to pheromone information, ACO algorithms typically use heuristic information to guide ants towards promising solutions. This heuristic information is usually denoted by η.

A generic ACO algorithm is described below as an illustration of the operation of the ACO metaheuristic. It is also given in Algorithm 2. $\mathfrak{S}_{active}$ represents the set of sequences that are yet to be completed, $\mathfrak{S}_{iter}$ represents the set of sequences constructed in each iteration, $\hat{\mathfrak{s}}$ represents the best solution produced at each iteration, and $\mathfrak{s}^*$ represents the best solution produced overall.

At the commencement of the algorithm's run, all pheromone values are set to the same level τ_0.[4] A set of m ants is created with each ant's solution either empty or consisting of one, possibly randomly chosen, component. Ants then construct solutions simultaneously until all ants have a complete solution. At each step, an ant selects a component to add to its solution probabilistically based on the relative pheromone intensity on available components and possibly some form of heuristic information. Pheromone may be updated at each step (called a *local pheromone update*), at the completion of a construction iteration (called *global pheromone update*), or a combination of the two. In the global pheromone update, ants retrace the path they took and deposit pheromone in inverse proportion to the quality of the solution they constructed. This simulates a larger number of ants being able to traverse a shorter path (i.e., better solution) than a longer path given the the same amount of time. All pheromone values are then decreased slightly in an analogue of evaporation. A number of alternative pheromone update schemes are described in the next section.

2.3 ACO Algorithms

A number of variations of the ACO metaheuristic have been proposed. This section provides a brief introduction to some of the more notable examples to illustrate the underlying mechanics of the ACO approach. Only fundamental features of the various approaches are covered as subsequent chapters consider features of the approach that are to a large extent independent of the specific features of actual ACO algorithms.

2.3.1 Ant System

Ant System (AS) evolved from the best-performing of a number of alternative ant algorithms developed by Dorigo et al. (1991). AS's application to the TSP is described to illustrate its operation. Solutions are constructed in the manner described in Section 2.1, i.e., ants create a permutation of cities by choosing one city to add to their respective partial sequences at each step. A TSP instance consists of a set of cities $\mathfrak{C}$ of size n, with known distances between them. The distance between cities i and j is denoted $d(i, j)$.

[4]Early ACO algorithms set τ_0 to a low value, while variants such as $\mathcal{MAX} - \mathcal{MIN}$ Ant System (Stützle and Hoos, 1996) (described in Section 2.3.3) set τ_0 to an upper bound imposed on pheromone values.

Algorithm 2 Ant Colony Optimisation metaheuristic

for all τ_i **do**
  $\tau_i \leftarrow \tau_0$
end for
while stopping criterion not met **do**
  $Active \leftarrow \{1, \ldots, m\}$
  for all $k \in Active$ **do**
    $\mathfrak{s}_k \leftarrow \langle\rangle$
  end for
  while $Active \neq \varnothing$ **do**
    for all $k \in$ Active **do**
      $\mathfrak{c} \leftarrow$ SelectComponentProbabilistically$(\mathfrak{N}(\mathfrak{s}_k), \tau, \eta)$
      $\mathfrak{s}_k \leftarrow \mathfrak{s}_k + \mathfrak{c}$
      if $\mathfrak{s}_k$ is complete **or** cannot be completed **then**
        $Active \leftarrow Active \setminus \{k\}$
      end if
      UpdateLocalPheromone$(\mathfrak{S}_{active}, \tau)$ $\{Optional\}$
    end for
  end while
  $\hat{\mathfrak{s}} \leftarrow$ BestSolutionConstructed$(\mathfrak{S}_{iter})$
  if first iteration **or** BetterCost$(\hat{\mathfrak{s}}, \mathfrak{s}^*)$ **then**
    $\mathfrak{s}^* \leftarrow \hat{\mathfrak{s}}$
  end if
  UpdateGlobalPheromone$(\tau, \mathfrak{S}_{iter}, \hat{\mathfrak{s}}, \mathfrak{s}^*)$
end while
Display $s_{\mathfrak{s}^*}$

Given that the objective of the TSP is to devise a Hamiltonian cycle of the cities such that the total distance travelled is minimal, a common choice for heuristic information is the inverse of the distances between cities, denoted $\eta(i,j) = \frac{1}{d(i,j)}$. It is also usual practice to associate pheromone with the links between components and hence the pheromone on the link between cities i and j is denoted $\tau(i,j)$. As all cities must be visited, it is typical that each iteration begins by adding a single randomly chosen city to each ant's solution sequence (although ants could in principle start with empty sequences). Then, at step t of the current iteration, ant k chooses its next solution component $\mathfrak{c}$ (i.e., city) probabilistically according to

$$p_k(t,\mathfrak{c}) = \begin{cases} \dfrac{\tau(\mathfrak{s}_k[t-1],\mathfrak{c})^\alpha \cdot \eta(\mathfrak{s}_k[t-1],\mathfrak{c})^\beta}{\sum_{\mathfrak{c}' \in \mathfrak{N}(\mathfrak{s}_k)} \tau(\mathfrak{s}_k[t-1],\mathfrak{c}')^\alpha \cdot \eta(\mathfrak{s}_k[t-1],\mathfrak{c}')]^\beta} & \text{if } \mathfrak{c} \in \mathfrak{N}(\mathfrak{s}_k) \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $p_k(t, \mathfrak{c})$ is the probability of ant k choosing component $\mathfrak{c}$ at step t, $\mathfrak{s}_k[t-1]$ is the last component added to ant k's solution sequence, and the parameters α and β control the relative importance of pheromone and heuristic information respectively. $\mathfrak{N}(\mathfrak{s}_k)$ is the set of unvisited cities.

Once all ants have constructed a solution, ants' solution sequences are used to update pheromone according to

$$\tau(\mathfrak{c}_i, \mathfrak{c}_j) \leftarrow \rho \cdot \tau(\mathfrak{c}_i, \mathfrak{c}_j) + \sum_{k=1}^{m} \Delta\tau_k(\mathfrak{c}_i, \mathfrak{c}_j) \tag{2.2}$$

where $\rho \in [0, 1]$ is the pheromone persistence (i.e., the evaporation rate is $1 - \rho$), and $\Delta\tau_k(\mathfrak{c}_i, \mathfrak{c}_j)$ represents ant k's contribution to pheromone on that edge, given by Q/L_k, where Q is an additional parameter of the algorithm.[5]

2.3.2 Ant-Q and Ant Colony System

Gambardella and Dorigo (1995) describe a family of ant algorithms called Ant-Q, based on a combination of AS and the reinforcement learning (RL) approach of Q-Learning (Watkins, 1989; Watkins and Dayan, 1992). While a number of alternative algorithms were developed, one showed clear improvements over the original AS. This algorithm differs from AS in two main respects. First, it uses a different component selection rule that favours making a greedy choice of the next component, called the *pseudo-random proportional* rule. Applying this rule to the TSP, ant k chooses its next component $\mathfrak{c}$ according to

$$\mathfrak{c} = \begin{cases} \arg\max_{\mathfrak{c}' \in \mathfrak{N}(\mathfrak{s}_k)} \left\{ \tau(\mathfrak{s}_k[t-1], \mathfrak{c}')^\alpha \cdot \eta(\mathfrak{s}_k[t-1], \mathfrak{c}')^\beta \right\} & \text{if } q \leq q_0 \\ j & \text{otherwise} \end{cases} \tag{2.3}$$

where $q \in [0, 1]$ is a uniform random number, $q_0 \in [0, 1]$ is a parameter controlling the probability of making a greedy choice, and $j \in \mathfrak{N}(\mathfrak{s}_k)$ is a component chosen randomly according to the probability distribution defined by Equation 2.1.

The second main difference is the way in which pheromone is updated. Ant-Q includes a local pheromone update—a feature of some early prototypes of AS—as well as a global pheromone update. Further, while all ants take part in local pheromone updates, Ant-Q allows only one ant's solution to be used for the global pheromone update, which can either be the *iteration best* or the *global best* (i.e., the best solution produced during the algorithm's run). Early applications of Ant-Q used the iteration best solution. The rationale for allowing only one good solution to update pheromone values is to encourage later ants to search in the neighbourhood of that best solution (Bonabeau et al., 1999).

[5] The value of Q has little impact on the the performance of AS for $Q \geq 1$ (Dorigo et al., 1991).

As a refinement of their work on AS and Ant-Q, Gambardella and Dorigo (1996) and Dorigo and Gambardella (1997b) developed the Ant Colony System (ACS), which is used more widely than its predecessor. The major differences in ACS are the absence of the parameter α from Equation 2.3 (as $\alpha = 1$ was generally found to give good results), and the use of the global best solution to update pheromone, which can make the algorithm very greedy (Stützle and Hoos, 2000).[6] The local pheromone update rule of ACS, which is highly similar to that in Ant-Q, is given by

$$\tau(\mathfrak{s}_k[t-1], \mathfrak{s}_k[t]) \leftarrow (1-\gamma) \cdot \tau(\mathfrak{s}_k[t-1], \mathfrak{s}_k[t]) + \gamma \cdot \tau_0 \tag{2.4}$$

where γ is the proportion of pheromone to remove (to discourage ants in the same iteration from taking the same path) and τ_0 is the initial amount of pheromone deposited on each component. For the TSP, Gambardella and Dorigo (1996) use $\tau_0 = \frac{1}{n \cdot L_{nn}}$, where L_{nn} is the length of the solution generated by a nearest neighbour heuristic (see Reinelt (1994) for a description of this heuristic).[7]

ACS was one of the first ACO algorithms to make use of local search techniques to improve the solutions generated by ants, as well as to use candidate sets to reduce the number of components at each step allowing its application to large problem instances. Both of these features are discussed in more detail in Section 2.4.2 below.

2.3.3 $\mathcal{MAX} - \mathcal{MIN}$ Ant System

As ACO is an autocatalytic process, there is the risk that relatively high pheromone levels on certain solution components may quickly lead to those components being used to the exclusion of all others, resulting in premature convergence to a suboptimal solution.[8] This problem was especially noticeable on large TSP instances, motivating the development of the $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) (Stützle and Hoos, 1996, 1998, 2000).

The distinguishing characteristic of $\mathcal{MM}$AS is its imposition of bounds on pheromone values such that all values fall in the range $[\tau_{min}, \tau_{max}]$. Additionally, pheromone levels are initially set to τ_{max} to encourage exploration early in the search process (Stützle and Hoos, 2000). Stützle and Hoos found that the best solutions were found when the search was close to stagnation, and used this to guide the selection of values for τ_{min} and τ_{max}. This makes the selection of appropriate bounds rather problem specific. Details for the TSP are given

[6] See Gambardella and Dorigo (1996) and Dorigo and Gambardella (1997b) for a more detailed discussion of the differences between the two algorithms.

[7] The rationale for using L_{nn} is that it gives a rough approximation of the optimal solution, while taking the inverse of this value after multiplying by n gives a small value that is dependent on problem size (M. Dorigo, personal communication, 4 April, 2002).

[8] This phenomenon is also observed in real ant colonies, where a colony is unable to exploit a new, short path due to pheromone saturation of existing longer routes (Bonabeau et al., 1999).

in Stützle and Hoos (1996, 1998, 2000). Improvements to $\mathcal{MM}$AS include the use of static candidate sets and the addition of local search heuristics (Stützle and Hoos, 1998; Stützle and Hoos, 2000). Some $\mathcal{MM}$AS implementations also use a diversification mechanism to force the discovery of new solutions away from the global-best solution. This mechanism typically involves reinitialisation of all pheromone values to τ_{max} when little change in the solutions produced over time is detected. $\mathcal{MM}$AS has had a great deal of success across a range of problems (e.g., den Besten, Stützle and Dorigo, 2000; Ducatelle and Levine, 2004; Fenet and Solnon, 2003; Lourenço and Serra, 2002; Stützle and Linke, 2002).

2.3.4 Rank-based Ant System

Another improvement on the original AS is the Rank-based Ant System (AS_{rank}) of Bullnheimer, Hartl and Strauss (1999b). AS_{rank} is an extension of work involving AS with *elitist ants*, first introduced by Dorigo et al. (1991). Using the elitist ants strategy, the best solution produced at each iteration is further reinforced as if σ ants had produced it. This approach leads to improvements in solution quality, but is less effective as the differences between ants' solutions become smaller. The introduction of ranking helps in such situations. The combination of elitist ants and ranking of ant solutions is AS_{rank}. At the end of each iteration, ants are ranked according to the quality of their solutions. The best solution produces the largest change to pheromone levels, while the next ω solutions update pheromone in linearly decreasing amounts according to their rank.

2.3.5 Hyper-cube Framework for ACO

The hyper-cube framework (HCF) for ACO is not an alternative ACO algorithm so much as an alternative approach to the use and updating of pheromone values (Blum, 2004; Blum and Dorigo, 2003; Blum, Roli and Dorigo, 2001). The main characteristic of this approach is that pheromone values are bounded between 0 and 1, with associated changes to the standard equations used to deal with pheromone. While this is similar to the approach of $\mathcal{MM}$AS, τ_{max} is typically not bounded at 1, but derived based on the problem being solved. The HCF is based on the approach of modelling a COP by a set of binary decision variables. That is, each variable can receive the value 0 or 1. While pheromone information is typically used with problems where the decision variables appear to have $n > 2$ possible values, each individual pheromone value actually relates to a single solution feature—a feature that a solution may or may not exhibit. Seen from this perspective, pheromone values in the range $[0, 1]$ are simply a relaxation of a set of binary decision variables. For

an AS algorithm in the HCF, an individual pheromone value $\tau(i)$ is updated according to

$$\tau(i) \leftarrow \rho \cdot \tau(i) + (1 - \rho) \cdot \sum_{\{s \in S_{upd} | i \in s\}} \frac{F(s)}{\sum_{\{s' \in S_{upd} | i \in s'\}} F(s')} \tag{2.5}$$

where i is some identifiable characteristic of a solution with which a pheromone value has been associated, s is a solution, S_{upd} is the set of solutions used to update pheromone, ρ controls the relative influence of the old pheromone value against that calculated using the solutions in S_{upd}, and $F(s)$ is a *quality function* that determines the amount of pheromone to deposit on i (cf. $\Delta\tau$ in Equation 2.2).

Blum (2004) suggests the benefits of the approach include it being more robust across problems with different objective function values (cf. the value used for $\Delta\tau$ in standard AS), and that it allows for improved implementation of intensification and diversification, techniques used to intensify a search around particular solution features and diversify a search into new spaces of solutions respectively.

2.3.6 Other Variants

There are numerous other variations on ACO, some of which are outlined here. Cordón, Fernández de Viana and Herrera (2002a, 2002b) and Cordón, Fernández de Viana, Herrera and Moreno (2000) propose the Best-Worst Ant System (BWAS), which incorporates features of the Evolutionary Computation technique of Population-Based Incremental Learning (PBIL) into AS and ACS. The main distinguishing features of this algorithm are its penalisation of the worst solution generated at each iteration by a reduction in pheromone on its constituent elements, mutation of pheromone values, and reinitialisation of pheromone values when stagnation in the search is detected. Removing pheromone from the worst solution at each iteration was found to be ineffective, a result supported by work by Montgomery and Randall (2002). Reinitialisation of pheromone values when stagnation occurs is a major feature of $\mathcal{MM}$AS (Stützle and Hoos, 1998).

Influenced by population-based metaheuristics such as GAs and other EAs, Guntsch and Middendorf (2002b, 2002a) describe a novel ACO algorithm that maintains a population of solutions, Population Based ACO (P-ACO). Solutions in the population are taken from solutions produced by ants at each iteration, with the population's size limited to k individuals. Each solution in the population contributes a fixed amount of pheromone to the components that it contains. When a solution is removed, the same fixed amount of pheromone is removed from its components. Consequently, pheromone values can be discretised in the range $[\tau_{init}, \tau_{max}]$, where τ_{init} is an arbitrary initial pheromone amount. This results in a relatively fast pheromone update. Typically the best solution produced at each iteration is added to the population. To ensure the population size of k, various

schemes may be adopted for selecting which solution to remove from the population.

2.4 ACO Applications

This section provides an overview of the many applications of ACO to different COPs. It illustrates the breadth of ACO's application and general characteristics of ACO algorithms for particular kinds of problem. Applications to dynamic optimisation problems are not discussed as these typically involve the addition of features on top of an existing approach to solving static versions of such problems. Other reviews of ACO applications can be found in Dorigo and Di Caro (1999), Dorigo et al. (1999), Dorigo and Stützle (2002, 2004) and Stützle and Dorigo (1999a, 1999b).

The applications are grouped into categories based on the type of problem being solved. The categories place similar approaches together, and resemble the categories used in contemporary reviews such as that of Dorigo and Stützle (2004). Within each category an archetypical problem is described in detail to illustrate the essential features of problems in that category. See Table 2.1 for a sample of ACO applications.

2.4.1 Subset Problems

Subset problems, discussed in Section 2.1, involve the selection of a subset of items from some larger set subject to certain constraints. This group of problems includes, but is not limited to, the multiple knapsack (MKP) (e.g., Petersen, 1967; Leguizamón and Michalewicz, 1999), set covering (SCP) (e.g., Fiorenzo Catalamo and Malucelli, 2001), set partitioning (SPP) (e.g., Maniezzo and Milandri, 2002), maximum clique (MCP) (e.g., Fenet and Solnon, 2003) and k-cardinality tree (KCTP) problems (e.g., Blum, 2002b). These problems are typically simple to describe, but are NP Hard (Fiorenzo Catalamo and Malucelli, 2001). In some cases even finding a feasible solution is extremely difficult, as is the case with the SPP (Maniezzo and Milandri, 2002). A formulation of the MKP is given to illustrate some of the features of these problems.

The MKP is typically posed as a budgeting problem (Petersen, 1967), where a subset of projects $J = \{1, 2, \ldots, n\}$ must be constructed such that total profit is maximised while limitations on m resources are respected. It can be formulated as

$$maximise \sum_{i=1}^{|\mathfrak{s}|} p(\mathfrak{s}[i])$$

Table 2.1: Sample of ACO applications.

Problem type/problem	Reference(s)
Subset	
Multiple knapsack	Leguizamón and Michalewicz (1999)
Set covering	Fiorenzo Catalamo and Malucelli (2001), Rahoual, Hadji and Bachelet (2002)
Set partitioning	Maniezzo and Milandri (2002)
Maximum clique	Fenet and Solnon (2003)
k-cardinality tree	Blum (2002b)
Permutation/routing	
Travelling salesman	Bullnheimer, Hartl and Strauss (1999b), Cordón, Fernández de Viana, Herrera and Moreno (2000), Dorigo, Maniezzo and Colorni (1991, 1996), Gambardella and Dorigo (1995), Dorigo and Gambardella (1997b), Guntsch and Middendorf (2002b), Stützle and Hoos (1996)
Vehicle routing	Bullnheimer, Hartl and Strauss (1997, 1999a), Doerner et al. (2002), Ellabib, Otman and Calamai (2002), Gambardella, Taillard and Agazzi (1999)
Sequential ordering	Gambardella and Dorigo (1997)
Scheduling	
Job shop	Blum and Sampels (2002a, 2004) Colorni, Dorigo, Maniezzo and Trubian (1994), van der Zwaan and Marques (1999)
Flow shop	Stützle (1998), T'kindt, Monmarché, Tercinet and Laügt (2002)
Open shop	Blum and Sampels (2002a, 2004)
Total tardiness	Bauer, Bullnheimer, Hartl and Strauss (1999), den Besten, Stützle and Dorigo (2000), Iredi, Merkle and Middendorf (2001), Merkle and Middendorf (2001)
Group shop	Blum and Sampels (2002a, 2004)
Assignment	
Quadratic assignment	Cordón, Fernández de Viana and Herrera (2002a), Dorigo, Maniezzo and Colorni (1996), Maniezzo (1999), Maniezzo and Colorni (1999), Stützle (1997), Taillard and Gambardella (1997)
Frequency assignment	Maniezzo and Carbonaro (2000)
Generalised assignment	Lourenço and Serra (2002), Randall (2004)
Graph colouring	Costa and Hertz (1997)
Bin packing, cutting stock	Ducatelle and Levine (2001, 2004)
University timetabling	Socha, Knowles and Sampels (2002)
Constraint satisfaction	
Maximum satisfiability	Roli, Blum and Dorigo (2001), Schoofs and Naudts (2000) Solnon (2000, 2002),
n-queens	Solnon (2000)
Car sequencing	Gottlieb, Puchta and Solnon (2003)
Others	
2D HP protein folding	Shmygelska, Aguirre-Hernández and Hoos (2002)
Bus driver scheduling	Forsyth and Wren (1997)
Network synthesis	Randall and Tonkes (2001)
Shortest common supersequence	Michel and Middendorf (1999)
Aircraft landing scheduling	Randall (2002b)

s.t.

$$\sum_{j=1}^{|\mathfrak{s}|} r(i, \mathfrak{s}[j]) \leq c(i) \qquad \forall i \in \{1, 2, \ldots, m\}$$

where $\mathfrak{s}$ is the solution, $\mathfrak{s}[j]$ is the j^{th} item included in the solution, $p(j)$ is the profit associated with including project j in the project mix, $r(i, j)$ is the amount of resource i consumed by project j, and $c(i)$ is the amount of resource i available. Throughout the remainder of the thesis the more general term *item* is used instead of the term project.

In all ACO algorithms for the mentioned problems, with the exception of the SPP, solution components represent the items that may be chosen to form the subset. Thus $\mathfrak{C}$ is the set of items.

ACO algorithms for the MKP (Leguizamón and Michalewicz, 1999) and SCP (Fiorenzo Catalano and Malucelli, 2001; Rahoual et al., 2002) associate pheromone with the components in $\mathfrak{C}$ (items in the MKP, subsets in the case of the SCP). Consequently, pheromone is used to learn the utility of having a component in a solution or not. Leguizamón and Michalewicz's algorithm for the MKP produced good solutions to a number of benchmark instances, outperforming an EA developed by the authors. Fiorenzo Catalamo and Malucelli's ACO algortihm for SCP did not perform as well as other approaches tested by the authors on a suite of benchmark instances. Rahoual et al.'s ACO algorithm for the SCP was tested on a number of benchmark instances and was rarely able to find the optimal solution. However, its results were improved considerably with the addition of a local search procedure to improve the solutions generated at each generation.

In the edge-weighted KCTP, which consists of finding a subtree with k edges in some graph such that the weight of included edges is minimal, $\mathfrak{C}$ represents the set of edges. An ACO algorithm for this problem examined the use of two pheromone representations (Blum, 2002b). The first associates pheromone with the edges chosen in the same way as pheromone for the MKP and SCP. The second is a *higher order pheromone representation* (Blum and Sampels, 2002a) that associates pheromone with pairs of edges that are copresent in a solution. The intention of this representation is to obtain greater information about interdependencies between edges. Higher order pheromone representations are discussed in more detail in Section 4.1. Blum found ACO using the first pheromone representation performed best and was also able to outperform a problem-specific heuristic, while ACO using the second pheromone representation did not perform as well although still outperformed the same problem-specific heuristic on some instances. Both versions of the ACO algorithm employed a local search procedure to improve the solutions generated by the ants.

The MCP involves selecting a subset of connected nodes from a graph (a clique) such that the size of the clique is maximal. Hence $\mathfrak{C}$ is the set of nodes. An ACO algorithm

for this problem also uses a higher order pheromone, associating pheromone with pairs of nodes to indicate which nodes should be added to a clique given those nodes already present (Fenet and Solnon, 2003). Thus, for both the KCTP and MCP pheromone has been associated with pairs of solution components being copresent in a solution. Fenet and Solnon's ACO algorithm, which does not employ a local search procedure, was able to find the optimal solution to a number of difficult benchmark instances, although its performance is not as good as the best performing specialised heuristic for the MCP.

A SPP consists of a set of constraints and a collection of subsets, each of which covers a different number of constraints and has an associated weight. The aim is to select a number of these subsets such that they form a partition of the set of constraints of minimal weight. This would suggest that solutions should be built from components from $\mathfrak{C}$ where $\mathfrak{C}$ is the set of subsets. However, this problem is highly constrained and constructing solutions in such a straightforward manner often results in infeasible solutions (Maniezzo and Milandri, 2002). Maniezzo and Milandri propose an ant-based tree search procedure (cf. the construction tree defined in Section 2.1) for this problem that takes a different approach. Rather than selecting subsets until a partition is formed, ants consider each constraint in turn (each level in the tree represents a constraint) and assign a subset to cover it. Pheromone is then associated with using a subset to cover a particular constraint in a similar manner to its use in assignment problems (described in Section 2.4.4 below). While the costs of solutions produced by this ACO algorithm were not as good as the best performing metaheuristic for this problem, a GA developed by Chu and Beasley (1998a), the approach was successful in that the ACO algorithm was able to find feasible solutions.

2.4.2 Permutation/Routing Problems

The TSP, described in Section 2.3.1, is the first problem ant algorithms were applied to due to the high degree of similarity between ants finding the shortest path to a food source and artificial ants finding the shortest Hamiltonian cycle in a graph. Consequently, each major ACO algorithm has been applied to either the symmetric TSP or asymmetric TSP (ATSP)[9] (Bullnheimer et al., 1999b; Cordón et al., 2000; Dorigo et al., 1991; Dorigo et al., 1996; Gambardella and Dorigo, 1995; Dorigo and Gambardella, 1997b; Guntsch and Middendorf, 2002b; Stützle and Hoos, 1996). Each of these applications follows the approach of the first AS for the TSP, with $\mathfrak{C}$ representing the cities that must be visited and pheromone being associated with the links between cities/components. A more detailed description of ACO algorithms for the TSP up to 1999 can be found in Stützle and Dorigo (1999b).

Early ACO algorithms for the TSP were unable to produce competitive results com-

[9] An ATSP has at least one edge (i, j) where $d(i, j) \neq d(j, i)$.

pared with tailored heuristics. While algorithms such as ACS, $\mathcal{MM}$AS and AS_{rank} produced improved results, the best results have been obtained when local search heuristics were used to improve the solutions produced at each iteration (Dorigo and Gambardella, 1997b; Stützle and Hoos, 2000; Bullnheimer et al., 1999b). Local search heuristics are typically problem specific, and a number have been described for the TSP (see, e.g., Johnson and McGeoch, 1997; Reinelt, 1994).

On large instances (of any problem) it can be computationally expensive to evaluate every available component's probability. To make large TSP instances amenable to solution by ACO various ACO algorithms have made use of *candidate sets* (also called candidate lists). A candidate set is a subset of the available components, chosen to reduce the number of components that must be considered and often consisting of components that appear to be good to include. In ACS for the TSP, statically generated candidate sets have been used where, for each city, a set of the cl closest cities is maintained. In effect, these separate sets of components represent a single set of candidate edges, but it is useful for the TSP to consider them separately. When constructing solutions, ants consider first only those components in the appropriate candidate set and, only if that is exhausted, do they examine the remaining components. The use of static candidate sets has allowed the application of ACS to TSP instances of more than 1000 cities (Dorigo and Gambardella, 1997a). The use of dynamically generated candidate sets (that take pheromone information into account) has also improved algorithm speed as well as achieving improvements in solution quality (Randall and Montgomery, 2002).

Another problem closely related to the TSP is the vehicle routing problem (VRP), which occurs commonly in the design of distribution networks such as the postal service (Bullnheimer, Hartl and Strauss, 1999a). A VRP consists of delivering commodities of different weights to a number of customers (cf. cities) using vehicles with specific capacities so as either minimise the number of vehicles used or the total distance travelled. An added complication is that vehicles must start at and return to a depot. Each vehicle also has a maximum distance/time for its route. Bullnheimer et al. (1999a) have developed a version of their AS_{rank} algorithm for this problem. They note that once customers have been assigned to vehicles, the VRP reduces to a number of TSPs. Hence, their algorithm is applied to the VRP in the same way as ACO algorithms for the TSP, with the following modification. When the current route makes it impossible to select another customer without violating a vehicle's capacity, or would exceed the vehicle's maximum route length, the next component chosen is the component representing the depot. Bullnheimer et al. and Doerner et al. (2002) have in their respective ACO algorithms found that using a more complex heuristic measure than the inverse of the distance between customers can lead to improved results. Both of these ACO algorithms make use of static candidate sets.

Bullnheimer et al.'s algorithm was able to find good solutions to a number of benchmark instances, although these solutions were not as good as those of the best performing algorithms for these problem, while Doerner et al.'s algorithm, which included sophisticated heuristic information, performed better than a number of alternative metaheuristics.

A variation of the VRP is the VRP with Time Windows (VRPTW) in which each customer has a time window during which it must be serviced. Gambardella, Taillard and Agazzi (1999) propose a dual colony ACO algorithm for solving the VRPTW with a hierarchical objective function, where the first objective is to minimise the number of vehicles used and the second is to minimise the total time. Unlike other ACO algorithms for the VRP, one depot for each vehicle is created so that pheromone linking customers to the depot does not become unduly high. This approach also allows the problem to be treated exactly like a standard TSP. This approach has also been used by Ellabib, Otman and Calamai (2002). Both algorithms were found to be competitive with a range of other metaheuristics for these problems.

The sequential ordering problem with precedence constraints (SOP) can be used in production planning (Gambardella and Dorigo, 1997). The problem consists of finding a Hamiltonian path (starting at node 0 and ending with node n) in a node and edge-weighted graph such that the weight of the included edges is minimised. This is similar to the machine scheduling problems discussed in the next section. The SOP can be modelled as an ATSP, with some additional precedence constraints, which is the approach taken in Gambardella and Dorigo's (1997, 2000) Hybrid Ant System for the SOP (HAS-SOP). HAS-SOP is called a hybrid AS because in addition to standard ACS it uses a local search heuristic to improve the solutions generated, which was a relatively novel feature of ACO algorithms when it was produced in 1997 (although now it is considered an integral part of ACO implementations). The ACS component of HAS-SOP is identical to ACS for the TSP except that $\mathfrak{N}(\mathfrak{s}^p)$ ensures that precedence constraints cannot be violated. The algorithm was able to outperform one of the leading heuristics for the SOP.

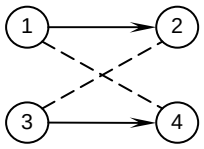
2.4.3 Scheduling Problems

The scheduling problems described in this section are described in terms of manufacturing, although they have wider applicability. In general, a scheduling problem involves a number of *operations* that must be performed by specific *machines* or agents in a particular order. Operations may belong to particular *jobs*, where operations from the same job cannot be processed at the same time. The number of machines and jobs varies between different kinds of scheduling problem. Different orders of operations result in different amounts of time to complete all operations, called the *makespan*. The typical aim of these problems is to minimise the makespan.

Problems in this group include the single machine total tardiness problem (SMTTP) (and variants) (e.g., Bauer et al., 1999; den Besten et al., 2000; Iredi et al., 2001; Merkle and Middendorf, 2001), flow shop (FSP) (e.g., Stützle, 1998), job shop (JSP) (e.g., Blum and Sampels, 2004; Colorni et al., 1994; van der Zwaan and Marques, 1999) and open shop (OSP) (Blum and Sampels, 2004) problems. The JSP, OSP and a generalisation of the two, are described in more detail to illustrate the features of these problems. An instance of either the JSP or OSP consists of a set of *operations* $\mathcal{O} = \{o_1, o_2, \ldots, o_{|\mathcal{O}|}\}$ partitioned into the *jobs* to which they belong $\mathcal{J} = \{J_1, J_2, \ldots, J_{|\mathcal{J}|}\}$ and the *machines* $\mathcal{M} = \{M_1, M_2, \ldots, M_{|\mathcal{M}|}\}$ on which they must be processed. Only one operation from a job may be processed at any given time, only one operation may use a machine at any given time and operations may not be preempted. In the JSP, precedence constraints impose a total ordering on the operations within each job (i.e., there is a fixed sequence in which operations must be processed), while operations may be processed in any order in the OSP. Each operation o_i has a non-negative processing time $p(o_i)$, and the aim of both problems is to minimise the makespan, which for a solution s is denoted by $C(s)$. Blum and Sampels (2002b) describe a generalisation of these problems where operations within each job are also partitioned into *groups* $\mathcal{G} = \{G_1, G_2, \ldots, G_{|\mathcal{G}|}\}$, with precedence constraints applying within groups. This generalisation is called the group shop scheduling problem (GSP), and is an important problem in the remainder of the thesis. In the JSP, each operation is assigned its own group (i.e., precedence constraints apply between operations), while in the OSP all operations within a job belong to a single group (i.e., there are no existing precedence constraints between operations). Given an existing JSP or OSP instance and adjusting the number, and hence size, of groups, a range of problem instances may be constructed with characteristics intermediate between the JSP and OSP.

It is common to represent instances of most permutation scheduling problems as disjunctive graphs,[10] where directed edges indicate existing precedence constraints (as exist in the JSP for instance) and undirected edges exist between operations that either require the same machine or are part of the same job but have no pre-existing precedence constraints between them. Operations connected by undirected edges are referred to as being *related* (Blum and Sampels, 2002a). Figure 2.5a depicts the disjunctive graph representation of a small JSP instance consisting of two jobs, both of two operations each. In this instance, operations 1 and 4 require the same machine, as do operations 2 and 3. A schedule for such problems may be created by assigning directions to undirected edges in the graph to create a directed acyclic graph. Each operation is then scheduled as early as

[10] A disjunctive graph is a directed graph that contains disjunctive edges. A disjunctive edge (i, j) is the superposition of an edge from i to j and an edge from j to i, where only one of the edges may be present in any non-disjunctive graph derived from the original disjunctive graph. Hence, a disjunctive edge is equivalent to a single undirected edge that must be assigned a direction.



(a)

s_1: $1 \prec 4,\ 2 \prec 3$ $C(s_1) = 60$

s_2: $1 \prec 4,\ 3 \prec 2$ $C(s_2) = 40$

s_3: $4 \prec 1,\ 3 \prec 2$ $C(s_3) = 60$

(b)

Figure 2.5: A JSP instance described by Blum and Sampels (2002b). a) A small JSP instance with $\mathcal{O} = \{1, 2, 3, 4\}$, $\mathcal{J} = \{J_1 = \{1, 2\}, J_2 = \{3, 4\}\}$, $1 \prec 2$, $3 \prec 4$, $\mathcal{M} = \{M_1 = \{1, 4\}, M_2 = \{2, 3\}\}$, $p(1) = p(4) = 10$, $p(2) = p(3) = 20$. $i \prec j$ indicates i must be processed before j. b) The three solutions to this problem described in terms of the relative order of operations that require the same machine.

possible given the precedence constraints imposed by this directed graph. The *list scheduler algorithm* is a constructive algorithm for these problems that ensures that cycles cannot be created in the disjunctive graph. It creates a permutation of the operations to be scheduled by successively choosing from those operations whose required predecessors have already been placed in the permutation. The relative order of related operations is determined by their relative positions in the permutation. Consequently, these problems are often called *permutation scheduling problems*. ACO algorithms for such problems use the list scheduler approach to construct solutions. Hence, $\mathcal{O} \equiv \mathfrak{C}$ and $\mathfrak{N}(\mathfrak{s}^p)$ contains only those operations whose required preceding operations already appear in $\mathfrak{s}^p$.

The JSP is the first permutation scheduling problem to which ACO was applied. Colorni, Dorigo, Maniezzo and Trubian (1994), influenced strongly by their work on the TSP, developed an ACO algorithm for the JSP. The construction graph they define is made up of the operations to be scheduled plus an additional node representing the empty sequence from which ants start. This extra node is required as pheromone is associated with the edges of the graph and, unlike the TSP, it is important to know which operation appears first in the schedule. Van der Zwaan and Marques (1999) describe an ACO algorithm similar to Colorni et al.'s, with only minor changes to the way pheromone is updated. Neither of these approaches was considered highly effective, obtaining results 8–9% worse than the optimal solution (Colorni et al., 1994; van der Zwaan and Marques, 1999).

Departing from the original TSP inspired approach, the majority of later ACO algorithms for scheduling problems associate pheromone with the absolute position of an operation in the permutation. This innovative approach removes the need for an artificial start node and appears better suited to describing permutations than associating pheromone with pairs of adjacent components. This approach has been used in ACO algorithms for the FSP (Stützle, 1998; T'kindt, Monmarché, Tercinet and Laügt, 2002) and single machine total tardiness problems (Bauer et al., 1999; den Besten et al., 2000; Merkle and Middendorf, 2000; Merkle and Middendorf, 2001), where the aim is to reduce the amount of time each job is late, and resource constrained project scheduling (RCPS) (Merkle, Middendorf

and Schmeck, 2000). In general this pheromone representation has been more successful than the TSP inspired pheromone used in initial applications. The latter approach is still used, however, such as in an ACO algorithm for a SMTTP with sequence-dependent setup times (Gagné, Price and Gravel, 2002). The two kinds of pheromone have also been used in combination in an ACO algorithm for the SMTTP with changeover costs (Iredi, Merkle and Middendorf, 2001). This version of the SMTTP has two objectives, to minimise job tardiness and to minimise costs associated with certain operations being scheduled immediately after certain other operations. Iredi et al. associate pheromone with the edges of the construction graph in order to minimise changeover costs, as these are associated with one operation immediately succeeding another. However, for minimising tardiness they associate pheromone with the absolute position of operations in the sequence.

To counteract some empirically observed problems with pheromone associated with the absolute position of operations in a solution sequence, Merkle and Middendorf (2000, 2002) suggest two alternative schemes for interpreting and updating pheromone values, called *summation evaluation* (Merkle and Middendorf, 2000) and *relative pheromone evaluation* (Merkle and Middendorf, 2002). Both schemes still associate pheromone with the absolute position of an operation in a sequence, but interpret and update the information differently. While the summing evaluation method was found to improve results compared to the standard way pheromone information is used, relative pheromone evaluation was found to produce the best performance. These approaches are discussed in more detail in Section 6.2.3.

Blum and Sampels (2002a) studied four different pheromone representations for the GSP: pheromone on edges of the construction graph; pheromone on the absolute position of operations in a sequence; pheromone on the absolute position with Merkle and Middendorf's (2000) summing evaluation; and a novel approach, pheromone on the precedence relationship established between pairs of related operations. The latter pheromone is associated only with pairs of related operations and is used to learn the utility of placing one of the operations before the other in the sequence, or more briefly, to *learn relations* (Blum and Sampels, 2002a). Hence it takes account of the dependencies between operations that require the same machine or which are part of the same job more directly than any of the other three pheromones. Blum and Sampels found that this pheromone outperformed the other three. Additionally, results for the other three pheromones support other findings in the literature, that associating pheromone with the edges of the construction graph for these problems generally performs worst, while associating pheromone with the absolute position of operations in a sequence is better, and is improved by the use of summing evaluation.

2.4.4 Assignment Type Problems

Assignment type problems (ATPs) involve the allocation of resources to items subject to a number of constraints (Costa and Hertz, 1997). Thus, the distinguishing characteristics of these problems are that they contain two distinct types of entity, items and resources, and that while all items must be assigned a (typically single) resource, the number of items assigned to one resource can vary from zero to many depending on the problem. This category of problems includes the generalised assignment (GAP) (e.g., Lourenço and Serra, 2002), quadratic assignment (QAP) (e.g., Stützle and Dorigo, 1999a), frequency assignment (FAP) (e.g., Maniezzo and Carbonaro, 2000), graph colouring (GCP) (e.g., Costa and Hertz, 1997), bin packing (BPP) and cutting stock (CStockP) (e.g., Ducatelle and Levine, 2004) problems. Due to their special features, ACO algorithms for the GCP, BPP and CStockP are discussed in the next section.

The GAP is described in more detail as it represents the archetypal assignment problem and is used as a key problem for later investigations in the thesis. The GAP is a resource allocation problem in which each of n tasks $\mathfrak{C}_{it}$ (i.e., items) must be assigned to exactly one of m agents $\mathfrak{C}_{res}$ (i.e., resources), where $m < n$ in general. Each possible assignment $(i, r) \in \mathfrak{C}_{it} \times \mathfrak{C}_{res}$ has an associated cost, and uses a certain amount of agent r's *capacity*. The aim of the problem is usually to minimise the total cost of assignments while respecting capacity constraints on agents.

In those ACO algorithms for which the construction graph has been described explicitly, ants typically move alternately between nodes representing items and nodes representing resources, as is depicted in Figure 2.6a. If the start node $\langle\rangle$ is removed, such a graph is consequently bipartite. All ACO algorithms for the GAP, QAP and FAP associate pheromone with the assignment of an item to a resource. In terms of this construction graph, this corresponds to associating pheromone with edges from items to resources. However, no pheromone is associated with edges leading back from resources to items, as such moves do not reflect a feature of the solution being constructed. Indeed, ACO algorithms for these problems typically determine an order in which to assign items and build solutions as sequences of resources, equivalent to the simpler construction graph depicted in Figure 2.6b. Consequently, components from $\mathfrak{C}_{it}$ are implicit in the sequences that ants construct from components in $\mathfrak{C}_{res}$. Costa and Hertz (1997) describe a general framework for solving ATPs where two pheromones are used, one to select the next item to be assigned and another to determine which resource to assign to which item. Assignment orders for the GAP and QAP are discussed in more detail in Section 3.3. ACO algorithms for the GAP (Lourenço and Serra, 2002; Randall, 2004) have taken this latter approach, and associate pheromone with the assignments made. The addition of a local search procedure improved results for both algorithms significantly. Lourenço and Serra found that their

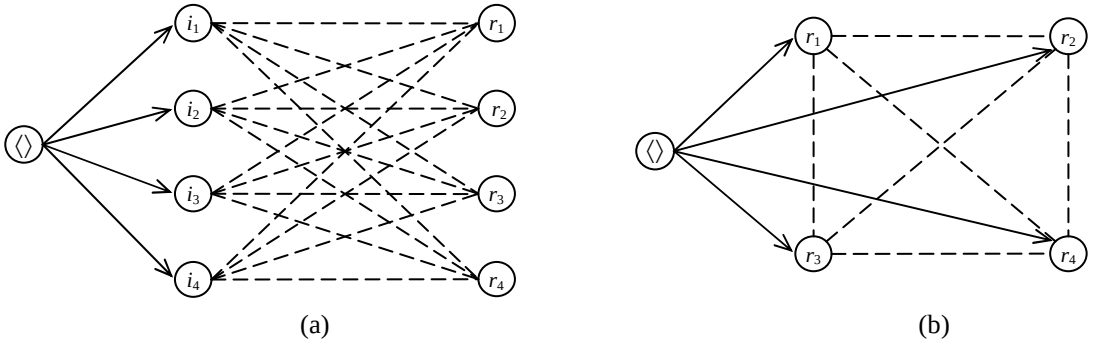


Figure 2.6: Two alternative construction graphs for assignment problems. The set of items $\mathfrak{C}_{it} = \{i_1, i_2, i_3, i_4\}$. The set of resources $\mathfrak{C}_{res} = \{r_1, r_2, r_3, r_4\}$. a) items appear as nodes in the graph. b) assignment order known, so only resources appear in graph. Dashed lines represent undirected edges that are implicitly assigned a direction as they are traversed by an ant.

ACO algorithm (with local search) was able to outperform a number of other metaheuristics for the GAP.

The QAP is a facility layout problem, with applications in building and office layout, keyboard design and scheduling (Stützle and Dorigo, 1999a). It consists of assigning n facilities (i.e., items) to n locations (i.e., resources), with exactly one facility assigned to each location. The aim is to minimise the total cost of flows between facilities (determined by a mixture of the distance between locations and the amount of flow required between facilities). Many ACO algorithms have been developed for this problem (e.g., Cordón et al., 2002a; Dorigo et al., 1996; Maniezzo, 1999; Maniezzo and Colorni, 1999; Stützle, 1997; Taillard and Gambardella, 1997). Many of these ACO algorithms produced results competitive with alternative solution approaches. Different assignment orders may affect the performance of ACO on this problem, and are discussed by Stützle and Dorigo (1999a) and also in Section 3.3. Several different kinds of heuristic information have also been developed, although they are not discussed further as they are not within the scope of this thesis. Additionally, some ACO algorithms for this problem have not used heuristic information (e.g., Stützle, 1997; Taillard and Gambardella, 1997).

Assignment of Items to Groups

Some assignment problems involve "resources" which are indistinguishable from one another and serve only to provide *groups* to which items can be assigned. It is convenient to refer to such problems as group assignment problems. Unlike the GAP, where two or more tasks may be assigned to the same agent and hence, appear to be in the same group, groups in these problems have no distinguishing characteristics, the important feature is which items are in the same group. The archetypical group assignment problem is graph

colouring. The GCP involves the assignment of colours to nodes in a graph such that no adjacent nodes are assigned the same colour. The actual colours assigned typically have no significance, serving only to create groups of nodes of the same colour. An assignment of colours to nodes that uses k colours is referred to as a k-colouring. The problem may be formulated either so that the number of colours used is minimised or that the number of colour conflicts (i.e., adjacent nodes with the same colour) is minimised for a given value of k.

Costa and Hertz (1997) describe two alternative ACO algorithms for the GCP, based on the order in which nodes are assigned colours. Both algorithms seek to minimise the number of colours required to create feasible colourings. The first variation constructs solutions by iteratively selecting the uncoloured node with the greatest number of colours assigned to its neighbours (i.e., adjacent nodes in the graph), and then assigning it either a colour from those already used in the partial solution or the next unused colour if all of the currently used colours would create a conflict. A construction graph for this approach would resemble that in Figure 2.6b. The second variation iterates through colour groups, assigning nodes to the current colour group until no more nodes can be assigned without creating a conflict, and then progressing to the next colour group. Nodes are examined in non-increasing order of degree. A construction graph for this approach would resemble that in Figure 2.6b with items and resources reversed. This second algorithm was able to outperform a number of other heuristic approaches for the GCP on large randomly generated graphs. Both variations associate pheromone with pairs of non-adjacent nodes assigned the same colour. The same approach is used in Ducatelle and Levine's (2001, 2004) ACO algorithms for the BPP and CStockP, where colour groups are replaced by bins and stocks respectively. Their algorithm for the CStockP outperformed a leading EA for that problem, while their algorithm for the BPP was less competitive against good heuristics for that problem.

Timetabling Problems

Timetabling problems often closely resemble the GCP, in that events (i.e., nodes, items) must be assigned to timeslots (i.e., colours, resources) without causing a clash (i.e., two events requiring the same participants being scheduled at the same time). Socha, Knowles and Sampels (2002) developed an ACO algorithm for a university course timetabling problem (UCTP) in which ants construct solutions by walking a construction graph of the kind depicted in Figure 2.6a. The UCTP being solved requires that only feasible timetables be created, with the aim of the problem being the minimisation of a number of soft constraints regarding good timetable design in terms of the students involved. They investigated the use of two different pheromone representations. The first associates pheromone with the

assignment of an event to a timeslot, in the same way as ACO algorithms for the GAP, FAP and QAP associate pheromone with the assignments made. The second pheromone is associated with pairs of events being assigned the same timeslot, as in the pheromone representation used with the GCP, BPP and CStockP. The second was expected to perform better, given the similarity between the GCP and timetabling problems in general. However, the first pheromone was found to produce better results, suggesting that it learns more effectively about features of solutions that affect the soft constraints which were being optimised. Using the first pheromone representation the algorithm was found to be competitive with alternative metaheuristics for this problem.

2.4.5 Constraint Satisfaction Problems

Constraint satisfaction problems (CSatPs) differ from typical COPs in that there is often no objective function to optimise, merely a set of constraints to satisfy. Commonly, they are formulated as maximal satisfaction problems, where the objective is to maximise the number of satisfied constraints.

One of the first applications of ACO to constraint satisfaction was Solnon's (2000) Ant-P-Solver for solving permutation satisfaction problems. Solutions to these problems are represented as a permutation of a set of values, where each position corresponds to the variable being assigned that value. Ant-P-Solver uses a construction graph in which ants move from an artificial start node to each value to be placed in the permutation in a similar way to ants constructing solutions to the QAP. However, the pheromone used is associated with the edges of this graph, in the same manner as ACO algorithms for the TSP and SOP. Ant-P-Solver achieves reasonable results on some problem instances. Gottlieb et al. (2003) use an ACO algorithm based on Ant-P-Solver to solve a car sequencing problem. The formulation of the car sequencing problem they solve is a constraint satisfaction problem in which a permutation of cars, each with a different set of *options*, must be devised such that the various stations along the production line can cope with the number of options of each type within a given amount of time. Gottlieb et al. found their ACO algorithm produced results comparable with a local search procedure.

More general constraint satisfaction problems have been tackled by Schoofs and Naudts (2000), Solnon (2002) and Roli, Blum and Dorigo (2001). These more general problems cannot be solved as permutations as each variable has its own domain which may overlap with other variables. All three approaches define a construction graph in which each node represents a variable–value pair, depicted in Figure 2.7. The ACO algorithms of Schoofs and Naudts and Solnon take a similar approach to those for the GAP. At each step, ants choose a variable to assign (various approaches to selecting the next variable are described by Schoofs and Naudts and Solnon). Next, ants use pheromone information to choose which

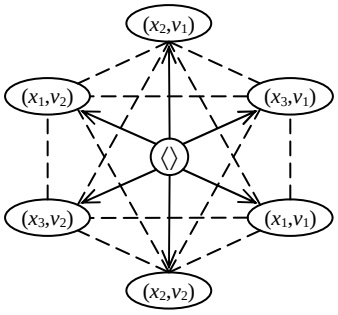


Figure 2.7: Alternative construction graph for assignment problems such as constraint satisfaction problems. In this example constraint satisfaction problem, the set of variables is $\mathfrak{C}_{it} = \{x_1, x_2, x_3\}$, while the set of values $\mathfrak{C}_{res} = \{v_1, v_2\}$. Dashed lines represent undirected edges that are implicitly assigned a direction as they are traversed by an ant.

value from that variable's domain to assign. In both algorithms, pheromone is associated with pairs of assignments. However, when using pheromone information to decide the value to assign to a variable i, Schoofs and Naudts only consider pheromone values for other variables already assigned that appear in the same constraints as i, while Solnon uses pheromone between all assignments currently made and the candidate assignment. Schoofs and Naudts found that their ACO algorithm outperformed a number of EAs for this problem. Solnon does not give comparative results for ACO and other metaheuristics, although did find that adding a local search procedure improved results.

Roli et al.'s (2001) ACO algorithm works somewhat differently, in that at each step any variable–value pair may be chosen provided that the variable in question is currently unassigned. Three different pheromone representations are considered: pheromone associated with individual assignments (i.e., the components in their specification); pheromone associated with one assignment being made immediately after another assignment (i.e., the links in the construction graph); and pheromone on pairs of assignments. The latter pheromone is effectively the same as that used by Solnon. Associating pheromone with either single assignments or collections of assignments performed well, while associating pheromone with links in the construction graph resulted in poor performance.

2.4.6 Other Problems

ACO has also been applied to a number of problems which do not fit easily into the categories already examined. However, the method of applying ACO in each of these has typically been similar to one of the approaches described above. The last two examples, ACO algorithms for the shortest common supersequence and for the scheduling of aircraft landings, are atypical in their respective approaches to pheromone usage and solution construction.

2D HP Protein Folding

Proteins are chains of amino-acids, with the function of a particular protein determined by the shape it takes when its chain folds up (Shmygelska, Aguirre-Hernández and Hoos, 2002). The shape of a protein when it is folded is difficult and expensive to determine, although the amino-acids in its chain may be more easily identified. Thus, being able to predict how a protein will fold and thereby determine its shape is an important problem in biochemistry. There are a number of models for predicting protein folding, one of which is the 2D Hydrophobic-Polar (2D HP) model. This model views amino-acids as belonging to either one of two groups, hydrophobic or polar, and represents a conformation (folding) of an amino-acid chain as a self-avoiding path on a 2D square lattice. The aim of this problem is to minimise the number of amino-acids labelled as hydrophobic that are adjacent on the lattice. Shmygelska et al. (2002) have developed an ACO algorithm for this problem (referred to as 2DHPPF hereafter) that begins solution construction by selecting a random amino-acid in the chain to be folded and placing it on a point in the grid. Rather than associate pheromone with the particular location in the grid to which an amino-acid is assigned, they associate pheromone with the relative change in direction of the chain from each amino-acid: left, right, or straight ahead. This takes account of the way the chain is folded regardless of how the folded structure may be rotated on the lattice. Consequently, they have transformed the problem into a type of assignment problem, where each amino-acid is assigned a direction to fold the rest of the chain from that point. Their algorithm produces some of the best results available for this problem.

Bus Driver Scheduling

Forsyth and Wren (1997) describes an ACO algorithm for scheduling bus drivers (the problem is referred to as BUS hereafter). The problem involves the assignment of drivers to periods of work such that the number of drivers required to cover all work is minimised, while respecting constraints related to the length of drivers' shifts, breaks for drivers between spells of work, and changeover times between drivers. Rather than use ACO to solve the entire problem, which is computationally intensive, a number of possible shifts are precomputed using another algorithm and ACO is used to select possible start times for a shift and then assign a shift that is available to start at that time. Thus the problem is modelled as an assignment problem, and pheromone is associated with the assignment of a shift to a particular starting time. This ACO algorithm was not found to be competitive with an existing system for the problem.

Network Synthesis

Telecommunication network synthesis (abbreviated to NETSYNTH hereafter) involves developing a network topology and bandwidth allocation between nodes for a telecommunication network (Berry, Murtagh, McMahon, Sugden and Welling, 1999). The only ACO algorithm for this kind of network synthesis problem, due to Randall and Tonkes (2001), does not use ants to explicitly solve either the topology or bandwidth allocation aspects of the problem. Instead, a set of routes (up to some maximum length) is precomputed, and it is from these that ants build their solutions. Bandwidth allocation is left to a subordinate heuristic. Rather than consider all possible routes, the proportion of routes to be randomly produced is a parameter of the algorithm. Pheromone is associated with the routes chosen to form each ant's solution. Consequently, the problem is solved in the same manner as knapsack problems. This ACO algorithm was able to find improved solutions compared to an alternative iterative metaheuristic developed by Randall (2000).

Shortest Common Supersequence Problem

The shortest common supersequence problem (SCSP) consists of creating a minimum length string of characters from some alphabet Σ such that it is a supersequence of a set L of other strings (i.e., any of the other strings may be produced by deleting characters from the solution) (Michel and Middendorf, 1999). The problem has various applications, for instance in the quite different fields of genetics and manufacturing, where Σ could represent genes on a chromosome or machines in a production line respectively. An ACO algorithm for this problem developed by Michel and Middendorf constructs solutions in the following way. Solutions are constructed from the characters in Σ, so $\Sigma \equiv \mathfrak{C}$. Furthermore, unlike the majority of ACO implementations, components in $\mathfrak{C}$ may—generally must— appear multiple times in a solution. Throughout the solution construction process, the algorithm keeps track of how many characters from the start of each string in L have been included in the partial supersequence. The *front* of each string in L is the next character that can be added to the supersequence. At each step, the set of candidate characters consists of the next character to include from each string. A pheromone value is associated with each character from the strings in L. However, the decision to include a candidate character $\mathfrak{c} \in \mathfrak{C}$ is based on a single pheromone value derived by summing pheromone values associated with next character in the strings in L where that character is $\mathfrak{c}$. Thus, a candidate character is more likely to be selected if it appears at the front of a relatively high number of strings and if the pheromone associated with that character at those positions in those strings is relatively high. This differs from more typical pheromone representations in that individual pheromone values have no meaning outside of the constructive process used as each will contribute to the inclusion of a character at some point in a single solution's

construction. In more typical pheromone representations the pheromone value associated with each solution component may be considered without that solution component being added to the solution.

Michel and Middendorf compared the ACO algorithm against a specialised heuristic and a GA, finding its performance was better than the specialised heuristic and comparable to that of the GA. With the addition of a lookahead function, the ACO algorithm was able to outperform the GA on some problem instances. The lookahead function heuristically estimates the utility of each candidate component by considering the quality of partial solutions that could be reached in the step after the current one if that candidate were chosen.

Aircraft landing scheduling

The static single runway aircraft landing problem (abbreviated to AIRLAND hereafter) involves the allocation of landing times to planes such that allocated times are within each plane's landing window while minimum separation times between different types of aircraft are respected (Randall, 2002b). Each plane has a preferred (most economical) landing time, and the aim of the problem is to minimise the difference between actual and preferred landing times, with different penalty rates for earliness and lateness. The problem can be solved as either a scheduling problem similar to the SMTTP, or as an assignment problem. If modelled as a scheduling problem, solutions could be sequences of planes, with a subordinate heuristic allocating planes as near to their preferred landing times as possible. Randall's ACO algorithm for this problem takes the second approach, with planes being considered in a randomised order for assignment of landing times. However, unlike the assignment problems considered in Section 2.4.4, where $|\mathfrak{C}_{it}| \geq |\mathfrak{C}_{res}|$, in this problem $|\mathfrak{C}_{it}| \ll |\mathfrak{C}_{res}|$. Furthermore, the differences between resources (i.e., timeslots) are gradual—each resource is not distinct as in the GAP and QAP, but similar to a number of other resources (nearby timeslots). Randall divides each plane's time window into a fixed number of contiguous regions, with pheromone associated with the assignment of a plane to a timeslot within a particular region. Therefore, while assignments are still made to individual timeslots, all timeslots in the chosen region receive an increase in their associated pheromone. The implications of this use of pheromone are discussed in Section 6.2.4.

Results of the ACO algorithm applied to a number of benchmark instances showed it to be competitive with a specialised heuristic developed by Beasley, Krishnamoorthy, Sharaiha and Abramson (2000).

2.5 Formalisations of ACO

Accompanying the maturation of the ACO field have been a number of efforts to formalise and analyse the algorithm. For instance, some have developed convergence proofs for specialised classes of ACO algorithms (e.g., Gutjahr, 2000, 2002; Meuleau and Dorigo, 2002; Stützle and Dorigo, 2002), while others have developed generalised forms of the algorithm (Gutjahr, 2000, 2002) and frameworks for describing and analysing the algorithm (Birattari et al., 2002). These last two are of particular relevance to this thesis and are discussed in more detail.

2.5.1 Graph-based Ant System

Gutjahr (2000, 2002) describes a theoretical generalised ACO algorithm called *Graph-Based Ant System* (GBAS). The GBAS literature was some of the first to use the term *construction graph* to describe the way ants may sequentially add solution components to produce a solution. In the GBAS, ants construct directed walks in a construction graph which represents, possibly in abstract form, a solution to the problem being solved. A problem-specific function Φ transforms walks in the construction graph into feasible solutions. According to Gutjahr, nodes may appear at most once in a walk and pheromone information is associated only with the links in the graph. This definition is far more strict than the description of ACO given in Section 2.2, and indeed in the rest of the ACO literature (e.g., Dorigo and Di Caro, 1999; Dorigo et al., 1996; Dorigo and Stützle, 2004). The GBAS achieves its generality by requiring that all problems be transformed into shortest path problems on a (construction) graph. Thus, the underlying ACO algorithm employed remains unchanged regardless of the problem to which it is applied, with only the construction graph specified for its use and the function Φ being problem-specific. GBAS is as yet only a theoretical system, and no attempt has been made to describe how the function Φ may be described and given to the system, a critical step in terms of implementation.

This approach is conceptually appealing, as it provides a framework into which all problems may be fitted with a uniform treatment of pheromone information. However, despite the convergence proofs Gutjahr develops for this algorithm, much of the optimisation literature suggests that solutions should be produced (or modified) in ways that are tailored to suit the problem in question (e.g., Birattari et al., 2002; Michalewicz, 1996). Moreover, much of the ACO literature suggests that pheromone representations should be chosen to match the problem, rather than the constructive algorithm used (Birattari et al., 2002).

2.5.2 Ant Programming

Birattari et al. (2002) describe a general framework for the description and analysis of a range of constructive optimisation algorithms (including ACO) called *ant programming*. *Ant programming* combines a number of ideas from the fields of dynamic programming, reinforcement learning and ACO. A central part of its approach is to consider the application of a constructive optimisation algorithm to a problem as a multi-stage decision process, a concept from dynamic programming. In terms of ACO, each decision corresponds to the addition of a solution component to a partial solution. Dynamic programming, and hence *ant programming*, place an emphasis on the *state* of a constructive process, where each state is equivalent to a partial solution to the problem being solved. In ACO, transitions between states occur when a component is selected and added to a partial solution (the previous state), to produce a more complete partial solution or complete solution (the next state). In the most general case, the states can be arranged in a tree, where each node corresponds to one state and transitions form the edges between states, equivalent to the construction tree described in Section 2.1.1.

The aim of an algorithm in the *ant programming* framework is to derive an *optimal control policy* (a concept borrowed from dynamic programming) that will guide the process through a number of states until a final state is reached that corresponds to the optimal solution. In other words, the state graph is a representation of the space of solutions, and presents an alternative form of the problem to be solved. In addition to this state graph, Birattari et al. (2002) describe a *representation* graph, which corresponds to the space of actual solutions to the problem being solved. While decisions made by a constructive optimisation algorithm concern the next state (partial solution) to move to given the current state, decisions are informed by the partial solution that each state represents. In particular, Birattari et al. point out that in ACO algorithms the pheromone representation used should typically correspond to the solutions represented rather than the sequences produced by the algorithm. *Ant programming* presents some important concepts that are useful in the remainder of this thesis. First is its analysis of the space of partial sequences, which is the subject of Chapters 3 and 5. Second is its emphasis on the role of the space of solutions (Birattari et al. use the term *representations*) in which ants must make their decisions, which is important when considering the nature of the pheromone representation that should be used with a problem. This issue is discussed in the next section, and then is the subject of Chapters 4, 5 and 6.

2.6 Divergence of Construction Graph and Pheromone Representation

Given that ACO is based on the foraging behaviour of ants—clearly a shortest path problem—it may seem natural and even essential that the same restriction be imposed on applications of ACO to various optimisation problems. Certainly, the development of a construction graph is considered of crucial importance in much of the ACO literature (e.g., Bauer et al., 1999; Colorni et al., 1994; Dorigo et al., 2000; Dorigo and Di Caro, 1999; Dorigo et al., 1996; Dorigo and Stützle, 2002; Fenet and Solnon, 2003; Gottlieb et al., 2003; Roli et al., 2001; Solnon, 2002; Socha et al., 2002; Stützle and Dorigo, 1999a). However, ACO algorithms have increasingly been applied to problems that show a marked divergence from classic shortest path problems like the TSP. Consequently, construction graphs for these problems can be highly contrived and may offer little assistance in determining an appropriate pheromone representation. Birattari et al. (2002) suggest that a suitable pheromone representation should be carefully chosen by considering the problem being solved and *not* the construction graph used to define the construction process. All these factors have led to the *ad hoc* application of several novel pheromone representations that do not associate pheromone with any obvious feature of the construction graph, some of which have performed poorly.

This pervasive emphasis on the use of construction graphs in ACO has the potential to misrepresent the role of the construction graph in the ACO metaheuristic. The construction graph for problems such as the TSP not only defines the solutions that may be built, but is also a good analogue of the environment in which real ants forage for food. However, modelling all problems using such a graph does not mean that artificial ants are necessarily solving the same kind of problem as real ants. Indeed, as the following examples illustrate, the construction graph is often used in ways that depart quite strongly from shortest path problems like the TSP and real ants' foraging behaviour.

The construction graph implicitly defined in ACO algorithms for subset problems such as the MKP, SCP and KCTP has $\mathfrak{C}$ as the set of items with L fully connecting the elements of $\mathfrak{C}$ (similar to Figure 2.4). Each node visited in a walk in this graph corresponds to the inclusion of that item in the solution. However, as Leguizamón and Michalewicz (1999) suggest, there is no real concept of a path in these problems. Hence, even if pheromone is associated with the items (an intuitive choice used in several ACO algorithms for these problems), considering these problems in a terms of a graph is quite artificial. Even in the application of ACO to the MCP, where the construction graph is identical to the graph from which nodes are selected to form a clique, Fenet and Solnon (2003) associate pheromone with edges between *all* pairs of nodes in the clique under construction. Thus,

the pheromone values associated with a solution do not correspond to a single path.

For assignment problems, a suitable construction graph could be defined as $G_C = (\mathfrak{C}_{it} \cup \mathfrak{C}_{res}, L = \{(\mathfrak{c}_i, \mathfrak{c}_j) | \mathfrak{c}_i \in \mathfrak{C}_{it}, \mathfrak{c}_j \in \mathfrak{C}_{res}\})$ depicted in Figure 2.6a. Indeed, this has previously been suggested by Stützle and Dorigo (1999a). However, for most assignment problems only edges in one direction in this graph represent assignments, and as it is with these that pheromone is associated, other edges in the graph are superfluous. In fact, the majority of ACO algorithms for assignment problems use a construction graph like that depicted in Figure 2.6b with the item being assigned implicit in the solution construction process rather than the construction graph.

ACO algorithms for constraint satisfaction, a special type of assignment problem, have used the construction graph $G_C = (\mathfrak{C} = \{(x_i \in X, d \in D_i\}, L = \{(\mathfrak{c}_i, \mathfrak{c}_j) | \mathfrak{c}_i, \mathfrak{c}_j \in \mathfrak{C}, \mathfrak{c}_i^1 \neq \mathfrak{c}_j^1\})$, where x_i is a variable, X is the set of all variables, d is a value from x_i's domain D_i, and $\mathfrak{c}_i^1$ is the first part of the couple represented by $\mathfrak{c}_i$. The best performing pheromone representations for these problems have associated pheromone either with the elements of $\mathfrak{C}$ (i.e., the assignments, as in pheromone used in most other assignment problems) or with pairs of assignments made, similar to the pheromone used for the MCP. In contrast, algorithms that associate pheromone with the edges of the construction graph traversed by ants perform rather poorly (Roli et al., 2001).

Costa and Hertz's (1997) ACO algorithm for the GCP, and Ducatelle and Levine's (2001, 2004) ACO algorithm for the BPP and CSP both associate pheromone with pairs of items being assigned to the same group (colour in the case of the GCP, bins and stocks in the cases of the BPP and CStockP respectively). While this pheromone representation is intuitively appropriate for these problems, it cannot be easily reconciled with walks in any construction graph that could be defined to represent how ants construct solutions to these problems.

Bauer et al. (1999) consider two alternative construction graphs for the SMTTP. The first is a GBAS inspired state-oriented construction graph, similar to that depicted in Figure 2.3, with pheromone associated with the edges linking solution states. However, this was not used due to its enormous size ($O(2^n)$ nodes when scheduling n operations), prompting the development of a greatly simplified construction graph like that used in ACO algorithms for other scheduling problems. As with many ACO algorithms for such scheduling problems, pheromone is associated with the absolute position of an operation in a solution sequence, a feature not directly related to the construction graph.

The ACO algorithm for a UCTP of Socha et al. (2002) is described in terms of a construction graph consisting of nodes for events (i.e., items) and nodes for times (i.e., resources). However, neither of the pheromone representations they consider (pheromone on assignments, or pheromone on pairs of events assigned the same time) relates to a

feature of the walks ants create in this construction graph to produce solutions.

From the above examples, it is evident that ACO has been applied to numerous problems that are not naturally described in terms of graphs. The construction graph therefore serves to describe how ants may build solutions, although it is often insufficient to describe the whole construction process as extra constraints must be used to ensure that ants make feasible walks in the construction graph. Thus there has been a divergence between the construction graph as a means of representing the construction process and the construction graph actually representing the space of solutions (for some problems such as the TSP they are identical). As the construction graph has become primarily a means to specify the constructive process, it is the pheromone representation that determines how solutions are modelled by the algorithm. Consequently, new pheromone representations have had to be developed for many problems as walks in the construction graph do not describe solutions adequately, or must be mapped to solutions.

2.6.1 Constructive Heuristics and Virtual Construction Graphs

Given that construction graphs typically only define the construction process, and often given little information about the pheromone representation used, it may be useful to define a simple unifying framework: the virtual construction graph. As shown in Figures 2.2 and 2.3, the sequence or solution space that any constructive algorithm explores *may* be represented by means of a graph. However, that does not mean that the solution construction process *must* be defined in terms of that graph. For instance, constructive metaheuristics such as GRASP (Feo and Resende, 1995) are typically not defined in terms of a construction graph.

An ACO algorithm and its associated virtual construction graph may be developed in the following way. As described in Section 2.1, the algorithm $\mathcal{A}$ defines the components $\mathfrak{C}$ from which solutions are built, the components that may be chosen at each step $\mathfrak{N}$ and the mapping from sequences of solution components to solutions $\mathfrak{S}^p$. It may be that a construction graph is used as part of the definitions of $\mathfrak{N}$ and $\mathfrak{S}^p$, but this is not required. Depending on the particular problem being solved, additional entities may also be defined, such as the set of items $\mathfrak{C}_{it}$ in an assignment problem. A pheromone representation can then be developed that relates aspects of the solutions or sequences produced to individual pheromone values that can be used to influence the selection of solution components. This aspect of applying ACO is the subject of Chapters 4 and 6.

Given $\mathcal{A}$, a virtual construction graph may be defined as $G_C^{virt} = (\mathfrak{C}^{virt}, L^{virt})$, where $\mathfrak{C}^{virt}$ and L^{virt} are sets of virtual constructive components and constructive links respectively. The definition of $\mathfrak{C}^{virt}$ depends on the way in which $\mathcal{A}$ views sequences and solutions. Define two (partial or complete) sequences $\mathfrak{s}_1^p$ and $\mathfrak{s}_2^p$ as *equivalent* in the constructive algo-

rithm $\mathcal{A}$, denoted $\equiv_{\mathcal{A}}$, if and only if $s_{\mathfrak{s}_1^p} = s_{\mathfrak{s}_2^p}$, $\mathfrak{N}(\mathfrak{s}_1^p) = \mathfrak{N}(\mathfrak{s}_2^p)$, $S_{\mathfrak{s}_1^p} = S_{\mathfrak{s}_2^p}$, and the probability of selecting any component in $\mathfrak{N}(\mathfrak{s}_1^p)$ is identical for both sequences. That is, two sequences are equivalent if they represent the same partial solution, may be augmented by the addition of the same solution components with the same probability, and can become the same complete solutions. For example, given sequences $\mathfrak{s}_1^p = \langle 1, 2, 3 \rangle$ and $\mathfrak{s}_1^p = \langle 1, 3, 2 \rangle$ in an ACO algorithm $\mathcal{A}$ for the MKP that associates pheromone with the items chosen, $\mathfrak{s}_1^p \equiv_{\mathcal{A}} \mathfrak{s}_2^p$. This can be seen in Figures 2.2 and 2.3. Given this definition of $\equiv_{\mathcal{A}}$, the set of all partial and complete solution sequences defined by $\mathcal{A}$ may be partitioned into subsets of equivalent sequences. Each of these subsets corresponds to an element of $\mathfrak{C}^{virt}$ (see Figure 2.3). The set of virtual (directed) links L^{virt} can then be defined as $L^{virt} = \{(\mathfrak{c}_i^{virt}, \mathfrak{c}_j^{virt}) \mid \mathfrak{c}_i^{virt}, \mathfrak{c}_j^{virt} \in \mathfrak{C}^{virt}$ such that a sequence in $\mathfrak{c}_j^{virt}$ can be reached from a sequence in $\mathfrak{c}_i^{virt}\}$.

Given this definition of a virtual construction graph, each virtual link corresponds to one step of the constructive algorithm in question. It follows that a pheromone value may be implicitly associated with each virtual link. Which pheromone value from the underlying pheromone representation is used depends on the algorithm's implicit definition of $\equiv_{\mathcal{A}}$. Similarly, heuristic information may also be implicitly associated with each link. The utility of a virtual construction graph is that it *can* be defined for any ACO algorithm, regardless of how that ACO algorithm's constructive process is defined, not that it *must* be defined. Consequently, its existence can justify the examination of ACO algorithms without the need to define a construction graph, nor the requirement that the pheromone representation be defined in terms of a construction graph. Neither of these are essential features of the constructive heuristics and ACO algorithms discussed in the remainder of the thesis.

2.7 Local Search in ACO

In general, the solutions built by constructive metaheuristics are improved by the application of a local search procedure (Dorigo and Stützle, 2002; Feo and Resende, 1995), and as many of the example ACO applications described in Section 2.4 demonstrate, the performance of ACO is better when local search is used to improve constructed solutions before pheromone is updated. Consequently, local search is now considered to be an integral part of ACO applications, with the ACO algorithm providing good starting points for local search (Dorigo and Stützle, 2002, 2004). However, local search procedures for the problems to which ACO may be applied are not studied in detail in this thesis, with the following rationale.

Local search forms the basis of iterative optimisation heuristics and metaheuristics, and so local search techniques for a wide range of COPs have been extensively studied. As a

result, powerful local search procedures are available for many problems to which ACO may be applied. However, comparatively little research has been conducted concerning the systematic adaptation of ACO to suit different problems. Given the poor performance of some ACO algorithms, there is clearly scope to improve the way ACO is applied to certain problems and to improve more generally the process by which it is adapted to suit new problems.

This thesis assumes that ACO is capable of producing good solutions in the absence of local search *if it is applied correctly*, a claim supported by results in the ACO literature (Dorigo and Stützle, 2004). If this is not the case then the utility of ACO as an optimisation technique would be called into question. Implicit in this assumption is the view that the addition of a powerful local search procedure is an unsatisfactory remedy for a poorly implemented ACO algorithm, and consequently that the existence of powerful local search techniques should not preclude investigation into improvements to the ACO algorithm.

In large part, analysis of the ACO algorithm and factors affecting its performance does not require the effects of local search to be considered. Nevertheless, analysis of alternative approaches to the application of ACO to the TSP and QAP, presented in Chapter 7, does include comparisons of the algorithm with and without local search. Extensions to this work that include the impact of local search are outlined in the conclusions in Section 8.3.

2.8 Summary

Constructive heuristics iteratively build solutions from problem-specific solution components. ACO is a constructive metaheuristic in which successive populations of artificial ants build solutions, guided by a model of the solutions that may be produced called a pheromone representation. Ants may also be guided by heuristic information. A number of ACO algorithms have been developed based on the original ACO algorithm Ant System, each with features that improve the performance of the approach as an optimisation technique.

ACO has been applied to a diverse range of problems, including subset, routing, scheduling and numerous assignment type problems. The pheromone representations used in these applications are equally diverse. As ACO has been applied to an increasing number of problems that differ from the original shortest path problem to which it was applied, a range of *ad hoc* pheromone representations have been developed. Although a construction graph is commonly considered an integral part of the definition of an ACO algorithm, many of these *ad hoc* applications have either not used one or used a pheromone representation that does not relate to the construction graph used.

Although many ACO algorithms for the same problem have used the same type of

pheromone representation, the literature currently lacks a standard language for describing pheromone representations. Furthermore, in those problems for which a range of pheromone representations have been used, it is clear that some pheromone representations result in better performance than others. These two issues, a language for describing pheromone representations and predicting which pheromone representation will produce the best results, are dealt with in subsequent chapters. However, before they can be discussed it is necessary to examine the constructive process in more detail, as it is within this framework that ACO algorithms operate. This is the subject of the next chapter.

Chapter 3

Bias in Constructive Heuristics

As described in Section 2.1, the space in which constructive approaches operate can be viewed in different ways. For a given problem, together with a constructive algorithm for solving that problem, one may look at the *sequence space* that the constructive process explores, which forms a tree, or alternatively at the graph of partial solutions. The constructive process itself may also be defined by means of a construction graph, as is common practice in the application of ACO. However, while the constructive process must be clearly defined for a given problem, the sequence and partial solution spaces are *emergent features* of the constructive process and the problem to which it is applied.

Of particular interest in this chapter is the topology of the space of sequences, described by the construction tree, and the mapping from sequences to solutions and the impact these may have on the probability of individual solutions being found by an undirected constructive algorithm. Deliberate sources of bias, such as heuristic and pheromone information, are largely excluded from consideration in this chapter, although their potential impacts are discussed in Section 3.6. The reaction of different pheromone representations to any underlying bias is covered in Chapter 5.

Section 3.1 describes the underlying sources of bias that may exist in any constructive algorithm while Sections 3.2 and 3.3 deal with specialised forms of these biases in problems that may have infeasible solutions or in which the topology of the search tree can be altered by the algorithm. Section 3.4 provides some examples of bias in constructive algorithms for different COPs. Section 3.5 discusses the relative impact of these biases as problem size grows. Section 3.6 describes some of the deliberately introduced biases in constructive algorithms that are excluded from the analyses in this chapter, and suggests some of their possible impacts.

3.1 Bias Inherent in the Constructive Process

Throughout this chapter the term *undirected* is used to indicate that the constructive algorithm in question makes each constructive decision probabilistically using a uniform random distribution over the components available at each step. Such an algorithm is hereafter referred to as $\text{ACO}_{\text{undir}}$ (i.e., undirected ACO). This is a necessary simplification to study the biases that occur at the lowest level of any constructive optimisation algorithm. Throughout the remainder of the thesis, the term *solution bias* is used to describe any bias that favours one solution over another.

In constructive optimisation algorithms, there are two fundamental ways in which bias may be introduced: the mapping from sequences to solutions and the existence of imbalances in the topology of the construction tree. Collectively these will be referred to by the term *constructed solution biases*.

Depending on the problem being solved and the nature of the solution components used, the mapping from sequences of solution components to solutions may not only be many-to-one, but also non-uniform. Consequently, some solutions may be overrepresented in the construction tree. Consider $\text{ACO}_{\text{undir}}$ applied to the JSP, in which solutions are represented as permutations of the operations to be scheduled. As solutions are uniquely described in terms of the relative order of operations that require the same machine or that are part of the same job, it is possible that the positions of some operations in the permutation may be exchanged without changing the solution represented. Consider the JSP depicted in Fig. 3.1, hereafter referred to as `jsp2-2`. There are three distinct solutions to this problem, yet six feasible sequences. Of these, four correspond to solution s_2, which accordingly appears to have a $66\frac{2}{3}\%$ probability of being discovered by $\text{ACO}_{\text{undir}}$, twice that expected if each distinct solution could be found with equal probability. This constitutes a *representation bias*.

Definition 1 A constructive algorithm $\mathcal{A}$ applied to a given combinatorial optimisation problem is said to have a *representation bias* if there exist two solutions s_1 and s_2, $s_1 \neq s_2$ such that $|\mathfrak{S}(s_1)| \neq |\mathfrak{S}(s_2)|$. If the probability of producing each sequence representing s_1 or s_2 is the same, $|\mathfrak{S}(s_1)| > |\mathfrak{S}(s_2)| \rightarrow P(s_1) > P(s_2)$, where $P(s)$ is the probability of $\text{ACO}_{\text{undir}}$ producing solution s. □

Figure 3.2 depicts the constructive paths corresponding to feasible solutions to the `jsp2-2` instance. Using $\text{ACO}_{\text{undir}}$, the probability of choosing a particular component at a given node in the tree is inversely proportional to the number of alternative components at that node. Consequently, sequences found on paths with fewer alternatives at each node are more likely to be discovered than those on paths with more alternatives. If there are no infeasible sequences defined by $\mathcal{A}$, then the degree of nodes within each level in

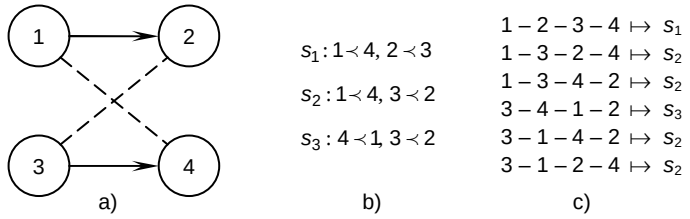


Figure 3.1: A JSP instance described by Blum and Sampels (2002b) a) A small JSP instance, `jsp2-2`, with $\mathcal{O} = \{1, 2, 3, 4\}$, $\mathcal{J} = \{J_1 = \{1, 2\}, J_2 = \{3, 4\}\}$, $1 \prec 2$, $3 \prec 4$, $\mathcal{M} = \{M_1 = \{1, 4\}, M_2 = \{2, 3\}\}$. $i \prec j$ indicates i must be processed before j. b) The three solutions to this problem described in terms of the relative order of operations that require the same machine. c) The six sequences that may be constructed and the solutions to which they correspond.

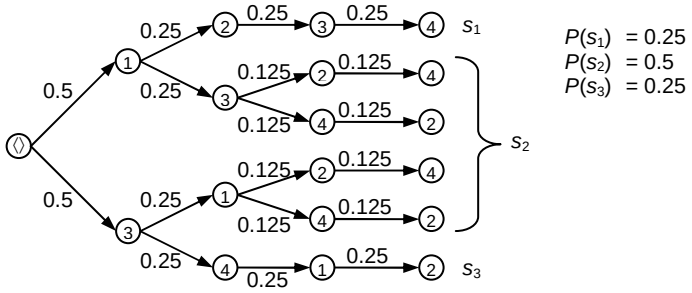


Figure 3.2: Construction tree for small JSP instance (see Figure 3.1 for problem description). Arcs are labelled with the probability of their being traversed if using $\text{ACO}_{\text{undir}}$. End points are labelled with the solution represented by that path. Aggregate solution probabilities appear on the right.

the tree will be uniform. This is the case, for example, in the TSP and QAP, where all permutations of cities or facilities represent feasible solutions. In problems where some sequences correspond to infeasible (or impossible) solutions, the degree of branching within each level will not be uniform, as in the JSP and GSP. In the example JSP, the probability of each of the sequences corresponding to solutions s_1 and s_3 is twice that for any of the four sequences corresponding to solution s_2. This constitutes a *construction bias*.

Definition 2 A construction tree $\mathcal{T}_{\mathcal{A}}$ has a *construction bias* if there exist two nodes in $\mathcal{T}_{\mathcal{A}}$ such that their heights in the tree are equal yet their degrees are not equal. A construction bias favours sequences on paths in $\mathcal{T}_{\mathcal{A}}$ with fewer alternative branches than other paths in $\mathcal{T}_{\mathcal{A}}$. □

When both biases are present, they will interact. Considering the `jsp2-2` instance, solution s_2 has $66\frac{2}{3}\%$ of all sequences. However, using $\text{ACO}_{\text{undir}}$, $P(s_2) = 0.5$. Hence, the distribution of paths in the tree corresponding to a solution's sequences determines the actual likelihood of constructing the solutions represented.

Different problems and construction approaches exhibit different combinations of the

two types of bias. The OSP is relatively unconstrained, with every permutation representing a feasible solution, so it exhibits only a representation bias. The more constrained GSP and JSP, both of which have sequences that cannot be constructed, have both a representation and construction bias. An alternative construction approach for the JSP, GSP and OSP which exhibits a different combination of bias is discussed in Section 3.4. In the TSP and QAP all sequences (assuming solutions are represented as permutations of either cities or facilities respectively) correspond to feasible solutions and each distinct solution has the same number of representations, so they have neither bias.

The example problems considered so far all have solutions of a fixed length. In problems where solutions are of variable length, or where feasible solutions cannot be guaranteed (and infeasible solutions terminate construction), the construction bias becomes more complicated.

3.2 Variable Length and Infeasible Solutions

For an unconstrained problem, the degree of branching at each decision point in the construction tree is determined by the size of the domain of values that each decision variable may take. The *constrainedness* of a problem under a constructive heuristic determines the amount of imbalance in this degree at each level of the tree. The construction tree for an unconstrained problem is generally perfectly balanced, with the same degree of branching on all paths. Problems with constraints that make certain solution representations impossible exhibit construction trees with imbalances; those paths on which decisions are made that restrict later options consequently have less branching than other paths.

In problems where construction paths are of variable length, the imbalance is not only evident in differences in the relative amount of branching on different paths, but also in their respective lengths. For instance, solutions to subset problems typically have different lengths depending on the components included in the solution and their respective impacts on the problem's constraints. For example, in the MKP a solution will generally be larger if made up of items with small resource requirements, while items requiring greater amounts of available resources will allow fewer other items in the solution. The early termination of a path also curtails opportunities for branching on that path, so shorter paths have an elevated probability of being reached. However, subset problems have a representation bias that favours larger solutions, and so these problems have two opposing biases. A lower bound on the probability of a sequence $\mathfrak{s}$ being produced by an undirected constructive search (denoted $P_{min}(\mathfrak{s})$), can be obtained by assuming the maximum amount of branching

on the path leading to $\mathfrak{s}$ and is given by

$$P_{min}(\mathfrak{s}) = \frac{(n-k)!}{n!} \tag{3.1}$$

where n is the number of available items in the subset problem and k is the number of items in $\mathfrak{s}$. An upper bound on the probability of a sequence $\mathfrak{s}$ being produced (denoted $P_{max}(\mathfrak{s})$) may be obtained by assuming the minimum amount of branching on the path leading to $\mathfrak{s}$, which occurs when the items in $\mathfrak{s}$ can only be included in solutions with other items in $\mathfrak{s}$. Consequently, all items are available in the first construction step, but only those items that appear in $\mathfrak{s}$ are available thereafter. The resultant upper bound is given by

$$P_{max}(\mathfrak{s}) = \frac{1}{n \cdot (k-1)!} \tag{3.2}$$

Given that there are $k!$ sequences corresponding to each solution of size k, lower and upper bounds on the probability of a solution s are given by Equations 3.3 and 3.4 respectively.[1]

$$P_{min}(s) = \frac{k!(n-k)!}{n!} \tag{3.3}$$

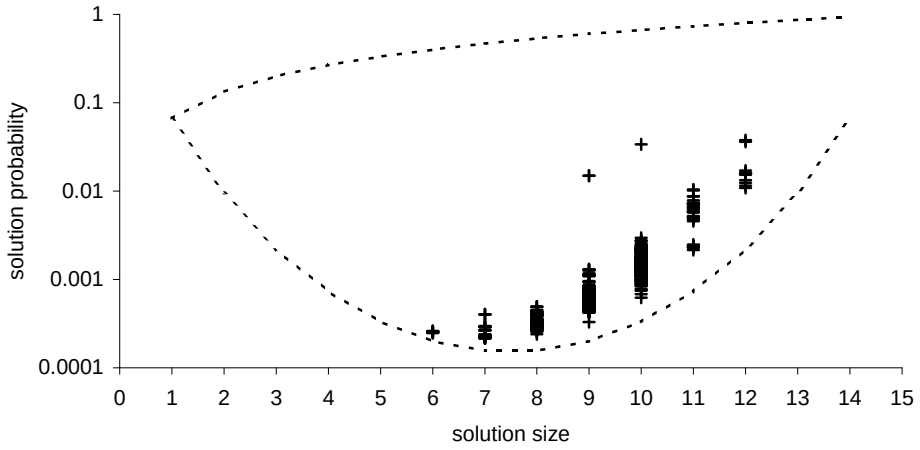
$$P_{max}(s) = \frac{k!}{n \cdot (k-1)!} = \frac{k}{n} \tag{3.4}$$

Figure 3.3 plots solution probability by size for the 15 item and 50 item MKP instances from the `mknap1` problem set available at the OR-Library (Beasley, 2005). Hereafter these instances are referred to as `mknap1-15item` and `mknap1-50item` respectively. The plot in Figure 3.3a shows results for all solutions to `mknap1-15item`, while that in Figure 3.3 shows the estimated probability given a sample of 300 randomly constructed sequences to `mknap1-50item`.[2] Lower and upper bounds are shown as dashed lines.

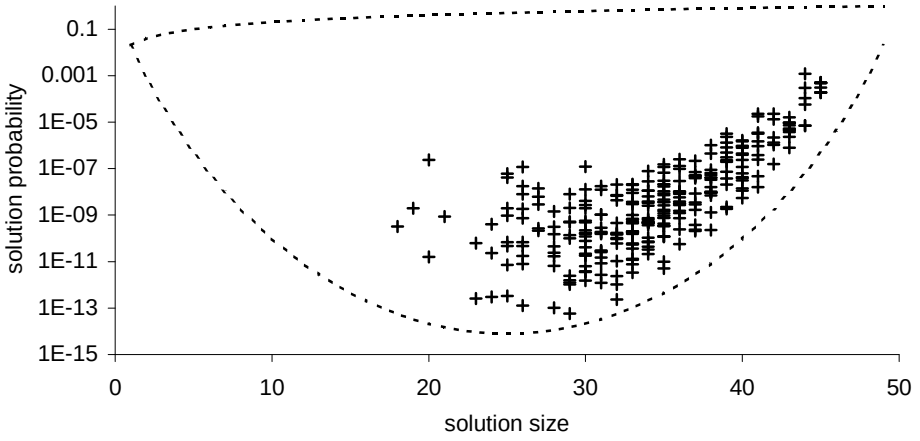
The lower bound on solution probability given by Equation 3.3 assumes the maximum amount of branching leading to each solution. Consequently, the interaction between the two sources of bias produces a minimum value for $P_{min}(s)$ when $k = \frac{n}{2}$. However, the existence of solutions with a range of sizes less than n (as a consequence of the existence of items that restrict the range of available possibilities when added to a partial solution) the amount of branching on each path to a solution will rarely be maximal. At the same

[1] Equation 3.4 may also be derived by considering just the initial construction step. Given a set of $k < n$ items that appear exclusively in a solution s, choosing any of those k (from n) items in the first step will necessarily lead to solution s. Consequently s has the highest probability of any solution of size k and $P(s) = \frac{k}{n}$.

[2] The probability of a solution $P(s)$ to `mknap1-15item` was estimated as $k! \cdot P(\mathfrak{s})$, where $\mathfrak{s}$ is one of the sequences representing s. This assumes each sequence's probability is indicative of all other sequences representing the same solution. Analysis of all sequences corresponding to a sample of solutions to `mknap1-15item` showed that in that problem the range of probabilities for sequences representing the same solution was less than 0.1% of the range $[P_{min}(\mathfrak{s}), P_{max}(\mathfrak{s})]$.



(a) 15 item MKP (`mknap1-15item`)



(b) 50 item MKP (`mknap1-50item`). Sample of 300 undirectly constructed solutions

Figure 3.3: Solution probability by size for `mknap1-15item` and `mknap1-50item` instances. Lower and upper bounds are shown as dashed lines.

time, the number of representations of each solution of size k remains $k!$. Consequently, the representation bias in this problem tends to dominate.

Variable length solution paths are also found in highly constrained problems where infeasible solutions cannot be avoided (and are not permitted by the algorithm in question and so necessarily halt solution construction). Such problems include the GAP and SPP. Techniques to avoid such solutions include backtracking, where a constructive algorithm may undo previous constructive steps, as well as allowing infeasible solutions to be completed before applying feasibility restoration heuristics, neither of which is the focus of this study. Backtracking is discussed by, e.g., Shmygelska et al. (2002), while feasibility restoration is discussed in relation to ACO by Randall (2002a) and Randall and Tonkes (2001). Lourenço and Serra (2002) describe an ACO algorithm for the GAP that permits infeasible solutions, but penalises them in proportion to the amount by which they violate constraints. In algorithms that do not admit infeasible solutions and which do not back-

track, such infeasible partial solutions have an elevated probability of being discovered. Furthermore, the more constrained a problem, the shorter will be the paths that lead to infeasible solutions, and so the relative increase in their respective probabilities will be even greater. This issue is discussed in more detail with regards to the GAP in Section 3.3.1 below.

Highly constrained problems are therefore very difficult for constructive heuristics to solve as not only are the majority of solutions in the solution space infeasible (e.g., this is the case in the GAP), but these infeasible solutions have a disproportionately high probability of being constructed. A number of such highly constrained problems are assignment problems, where the assignment order may alter the topology of the construction tree and hence the probability of producing infeasible solutions.

3.3 Assignment Problems and Assignment Order

In assignment type problems there exists a choice over the order in which items are assigned. While the construction tree is completely defined by the constructive process used to solve the TSP, MKP and GSP, changing the assignment order in an assignment problem rearranges paths in the tree. Different assignment orders do not alter the solutions represented as assignment problems typically have no representation bias. However, they will change which solutions are nearby in the construction tree and can introduce a bias in some problems. Figure 3.4 shows six alternative construction trees for a trivial GAP instance (2 agents, 3 tasks), each generated using a different, fixed assignment order. As Figure 3.4 illustrates, solutions that are found on nearby branches of the construction tree under one assignment order may be quite distant under another. For example, consider solutions s_2 and s_3, which share all but the last part of their paths in the construction tree in part (a), but whose paths diverge at the first decision in part (f). In effect, the assignment order determines the constructive neighbourhood in which solutions are found.

Consider a constructive heuristic for the QAP in which solutions are constructed by successively choosing an unassigned facility and then choosing a location to assign it. Given there is no representation bias nor are there any infeasible paths in the construction tree, rearranging solutions' paths in the tree by altering the assignment order does not alter solutions' respective probabilities. However, it does change which solutions are "neighbours" and hence may produce different results to another assignment order.

In contrast to the QAP, the GAP does have infeasible representations. Indeed, in many such problems there exist infeasible solutions that cannot be avoided *a priori*, a topic which is discussed in the next section. Considering those problem instances in which every partial solution can be completed, yet where there still exist infeasible solutions which

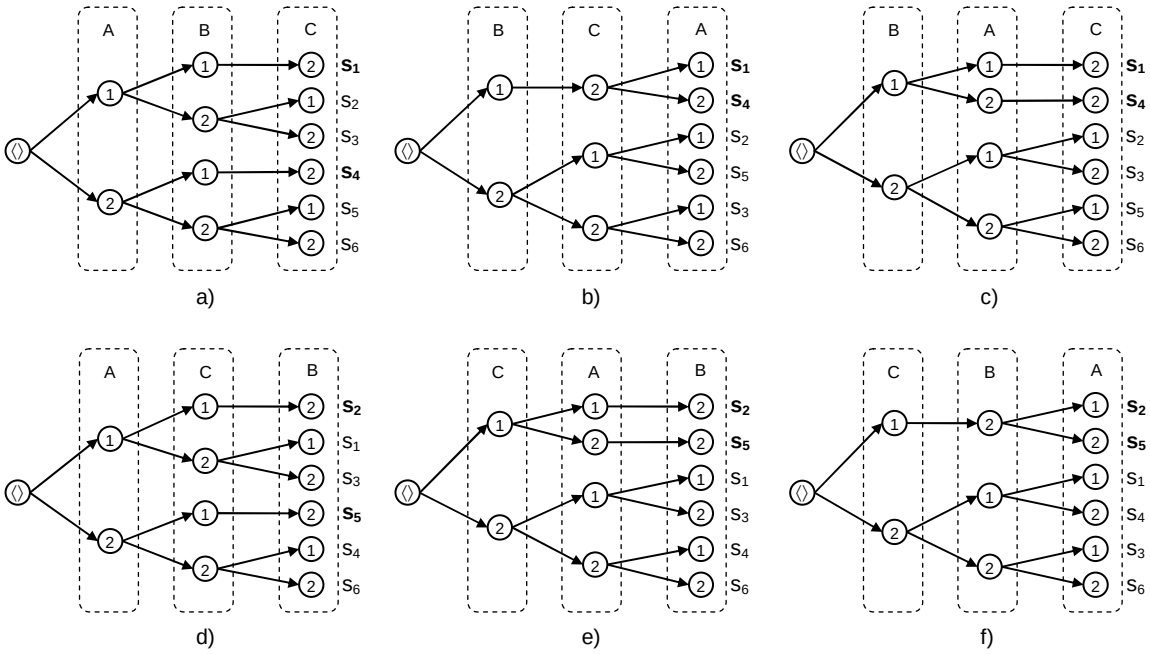


Figure 3.4: Construction trees induced by different assignment orders for small GAP instance with agents {1, 2} and tasks {A, B, C} in which the combined resource requirements of tasks B and C mean they cannot both be assigned to agent 1. Assignment orders are: (a) A–B–C; (b) B–C–A; (c) B–A–C; (d) A–C–B (e) C–A–B; and (f) C–B–A. The relative order of tasks B and C, which are competing for agent 1, determines which solutions are more likely; solutions s_1 and s_4 under assignment orders (a)–(c), solutions s_2 and s_5 under assignment orders (d)–(f). Solutions that are more likely appear in bold.

must be avoided, it follows that those solutions that lie near the bounds of feasible space will generally be favoured by $\text{ACO}_{\text{undir}}$ given they will be found on paths with relatively little branching. If decisions that push these solutions towards infeasibility are relatively near the initial empty solution, those solutions are more likely to be found. However, by altering the assignment order, such decisions may be moved to any level (i.e., constructive step) in the construction tree, thereby altering the degree of branching between the empty solution and the solutions affected by those decisions. Consequently, different assignment orders can alter the probabilities of different solutions being found by an undirected search.

Consider Figure 3.4, which depicts the construction trees induced by the six possible fixed orders of the three tasks involved. In this GAP instance, the combined resource requirements of tasks B and C exceed agent 1's capacity, so they cannot both be assigned to that agent, although all other combinations of assignments are feasible. Consequently, any path that assigns either B or C to agent 1 has a reduced number of options along its length, so solutions s_1, s_2, s_4 and s_5, each of which assigns either B or C to agent 1, can be found on paths with little branching under at least one of the assignment orders. Assignment orders (a)–(c) all have task B being assigned before task C, with the result that solutions s_1 and s_4 (both of which assign task B to agent 1) appear on paths with less branching and so are more likely to be found by an undirected constructive search. Assignment orders (d)–(f) reverse this relative ordering, so solutions s_2 and s_5 are more likely. In non-trivial instances, the interactions between the relative order of tasks that have conflicting requirements are far more complex, although the assignment order can still play a critical role in determining which solutions have a high probability.

Assignment orders may broadly be characterised by whether they are static (determined *a priori*) or dynamic (changed during solution construction) and whether they are randomly or heuristically determined. Which assignment order is "best" depends on both the type of assignment problem being solved and what the algorithm developer would like it to achieve. In problems with no infeasible solutions, such as the QAP, an assignment order is often chosen with the aim of improving solution quality. For instance, one ACO algorithm for the QAP uses a predetermined order in which facilities with high flow requirements are assigned early. Coupled with heuristic information that favours placing facilities in central locations (those with relatively short distances to all other locations) facilities that are likely to contribute to solution cost the most have an increased chance of being assigned to a central location which reduces their impact on cost (Maniezzo and Colorni, 1999). A number of other ACO algorithms select successive facilities randomly (Stützle and Hoos, 1998; Taillard and Gambardella, 1997). A dynamic, randomised assignment order is in effect a superposition of the construction trees produced by all other assignment orders. While this approach is not guaranteed to increase the likelihood of good solutions being

produced, it gives access to the construction trees for any other assignment order, and so over time allows the algorithm to explore a range of good and bad construction trees.

Of particular interest in this study is how assignment orders may be chosen for highly constrained assignment problems to either maximise the likelihood of reaching a feasible solution or at least reduce the infeasibility of solutions produced.

3.3.1 Using Assignment Order to Reduce Infeasible Space

Unavoidable infeasible solutions are a common feature of many assignment problems, where a number of items vie for limited resources. Devising an appropriate assignment order is thus an important part of solving these problems.

Using an arbitrary fixed order fixes the topology of the construction tree and solutions' probabilities. Choosing a random order implicitly assumes that the topology of the construction tree is unknowable,[3] and gives each possible construction tree topology an equal probability of being used. Intuitively, such an approach appears to give each item an equal chance of obtaining its most sought after resource. In the development of an ACO algorithm for the static aircraft landing problem, Randall (2002b) found that a dynamic random assignment order (of planes to landing timeslots) was more effective than an arbitrary fixed order. While randomised assignment orders will likely produce better results than an arbitrary fixed order, especially in an algorithm such as ACO which constructs a large number of solutions, they cannot guarantee a good outcome. An empirical investigation of assignment orders for the GAP (discussed in detail below) reveals that the majority of static, randomised assignment orders do not give a good probability of reaching feasible solutions in comparison with what is possible given problem constraints. Figure 3.5 shows the distribution of feasible solution probabilities achieved by a sample of 1000 random static orders for the five instances in the `gap2` problem set (available at the OR-Library (Beasley, 2005)).

Many ACO algorithms for assignment problems employ heuristics to determine a good assignment order. A commonly used heuristic is to sort items by non-increasing order of constrainedness (e.g., resource requirement in the GAP, number of shared constraints for a variable in the CSatP). This is often coupled with the use of heuristic information that biases ants towards assignments likely to not induce a constraint violation (e.g., Randall, 2004; Solnon, 2002). The rationale for this assignment order is that if such items are left unassigned until late in the construction process then there may not be sufficient resources to assign them. This heuristic would appear intuitively to produce a topology in which branching on paths that lead to infeasible solutions is maximised, as those items that are

[3]For non-trivial instances the topology of the construction tree is unknowable as it can only be revealed by complete enumeration of all solution sequences, which is an intractable problem for most COPs.

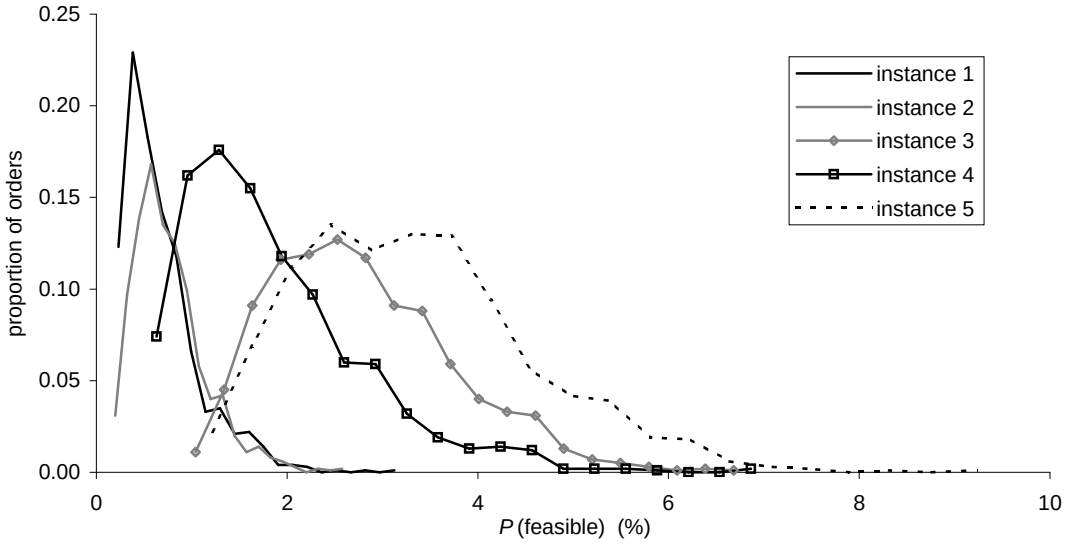


Figure 3.5: Distribution of feasible solution probabilities across 1000 assignment orders for `gap2` instances. Each distribution is divided into 20 regions to obtain points for the plot.

the most difficult to assign are assigned early, while the number of suitable resources for remaining items will still be high. An examination of different assignment orders for the GAP below shows that this assignment order produces construction trees with shorter paths leading to infeasible solutions, which consequently have an elevated probability relative to the proportion of paths they represent. However, the total number of paths leading to infeasible solutions is reduced, as decisions that lead to infeasibility are consolidated nearer the root of the construction tree. Although a small number of infeasible paths does not guarantee a high probability of reaching a feasible solution, trees with fewer infeasible paths typically have a higher probability of reaching a feasible solution than those with more infeasible paths. This is illustrated in Figure 3.6, which plots the number of infeasible paths against the probability of reaching a feasible solution for the same sample of 1000 static assignment orders described in Figure 3.5 applied to the first instance from the `gap2` set. Assignment order heuristics for a number of problems are now considered.

The GCP is an interesting assignment optimisation problem in that it may be posed in two alternative ways: either with the objective of minimising the number of colours required to produce a feasible colouring (i.e., where no two adjacent nodes have the same colour), or with the objective of minimising the number of colour conflicts (i.e., like-coloured adjacent nodes) for a given number of colours. While the former problem does not have infeasible solutions, the latter does. Costa and Hertz (1997) describe two alternative ACO algorithms to solve the first formulation of the problem, each of which assigns nodes using a different heuristically determined order. The first dynamically orders nodes in non-increasing order of saturation (the number of colours already assigned to a node's neighbours in the partial solution). The second iterates through colour groups, rather than nodes. However, when

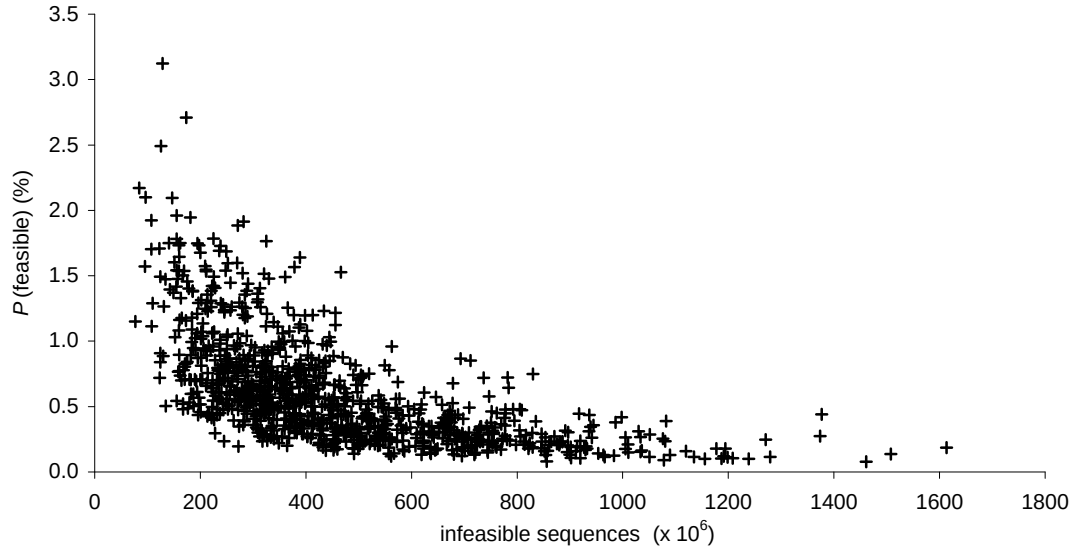


Figure 3.6: Number of infeasible sequence plotted against the probability of producing a feasible solution (denoted P(feasible)) for 1000 assignment orders for first instance from the `gap2` problem set.

selecting which node to add to a colour group nodes are ordered according to their degree. Costa and Hertz also discuss the potential use of static assignment orders including ordering nodes randomly or in non-increasing order of degree. Given the superior performance of the dynamic orders, they are the ones they incorporate into the ACO algorithms. Although the ACO algorithms developed by Costa and Hertz produce only feasible colourings, and hence the assignment orders they use appear designed to improve solution quality, they are also useful for solving the second formulation of the GCP in which a fixed number k of colours must be used with the aim of minimising the number of colour conflicts. Taking the first algorithm as an example, for a given number of colours k and a partial solution that uses k colours, if the next node must be assigned a colour outside the k already used then that step is equivalent to producing an infeasible partial solution if one were attempting to produce a k-colouring of the graph. Thus, heuristics that assign nodes based on some measure of how difficult they will be to colour will work for both formulations.

Constructive algorithms for constraint satisfaction problems often employ heuristics to determine a good order in which to assign the variables (Solnon, 2002). The ordering heuristics are distinguished by the way they measure the constrainedness of the variables. For example, commonly used orders include *most-constraining-first*, which selects an unassigned variable that appears in the greatest number of constraints with other unassigned variables, *most-constrained-first*, which selects an unassigned variable that appears in the greatest number of constraints with assigned variables, *smallest-domain*, which selects an unassigned variable whose domain is smallest once values that would lead to infeasibility are removed, and random ordering, which is the only one of the four not concerned with

assigning highly constrained variables early. Smallest-domain was used by Solnon's (2002) ACO algorithm for the CSatP.

ACO algorithms for the GAP also make use of ordering heuristics based on measures of the constrainedness of the tasks to be assigned (Lourenço and Serra, 2002; Randall, 2004). The properties of the construction trees induced by some of these kinds of orders are investigated empirically below.

In highly constrained assignment problems, a good construction tree topology is likely to be one in which the probability of discovering a feasible solution is maximised. Certainly, many of the ordering heuristics employed by constructive algorithms are designed for this purpose. However, finding an assignment order that produces such a tree is non-trivial. For a GAP with n items, there exist $n!$ alternative assignment orders and their corresponding construction trees. Moreover, to accurately assess the probability of producing a feasible solution, complete enumeration of each construction tree is required, which if practicable would subsume the original task of finding good solutions to the original GAP. As the set of all dynamic assignment orders is a superset of all static assignment orders, identifying a dynamic assignment order that maximises the probability of producing feasible solutions is even more difficult than identifying a static order with this property. Consequently, heuristically determined assignment orders are the only practical alternative.

An investigation of the probability of reaching a feasible solution using different assignment orders in the GAP is now described.

Case Study: Assignment Orders for the GAP

The preceding sections presented three main hypotheses regarding the number and probability of producing infeasible solutions under different assignment orders. Chiefly, these are that infeasible (and hence, relatively short) sequences have an elevated probability of being discovered, that assigning highly constrained items (tasks in the GAP) early produces shorter paths leading to infeasible solutions, and that assigning highly constrained items early should decrease the total number of such paths. This section presents an empirical investigation of these hypotheses and of the effectiveness (in terms of the probability of reaching a feasible solution) of a number of assignment orders when solving the GAP.

Initial analysis was made of the construction trees for every static assignment order for a trivial instance with 3 agents and 8 tasks (adapted from the first instance, with 5 agents and 15 tasks, in the `gap1` problem set).[4] With more than 99% of assignment orders, the probability of producing infeasible solutions was elevated above what would be expected given the total number of infeasible paths in the construction tree, in some cases

[4]Details of the 3 agent, 8 task instance are available in Appendix C, while the `gap1` problem set is available from the OR-Library (Beasley, 2005).

by more than 35%. The probability of reaching a *feasible* solution when the number of paths leading to infeasible solutions is minimal was found to be better than the probability under the worst assignment order (13% versus 3%). However, the highest probability (34%) was shown under another assignment order in which the number of such paths was low, but not minimal. Under this assignment order, the probability of producing infeasible solutions was *below* what would be expected given their number. Furthermore, the best assignment order did not have tasks in non-increasing order of constrainedness, which had a 16% probability of producing feasible solutions. However, as heuristic assignment orders that assign tasks in non-increasing order of constrainedness (or some variant thereof) are commonly used, they are the subject of the remainder of this investigation.

When ordering tasks based on their degree of constrainedness, a suitable measure must be used. Randall (2004) uses a dynamic measure of constrainedness based on the difference between each task's requirements and the current remaining capacity of each agent. The constrainedness $t(i)$ of a task i is given by

$$t(i) = \sum_{j=1}^{|\mathfrak{C}_{res}|} (curr_j + a_{ij}) - b_j \tag{3.5}$$

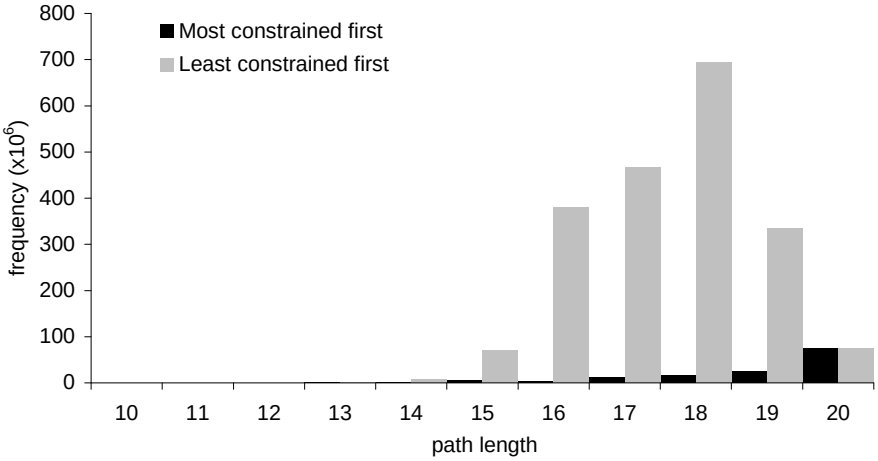
where $curr_j$ is the current amount of agent j's capacity in use, a_{ij} is the amount of agent j's capacity required by task i, and b_j is agent j's total capacity. The next task i to assign is selected from those unassigned tasks such that $t(i)$ is maximal. An alternative constrainedness measure was used in this investigation, based on the proportion of each agent's capacity a task would require, given by

$$t(i) = \sum_{j=1}^{|\mathfrak{C}_{res}|} a_{ij} / \max\{curr_j, nocap\} \tag{3.6}$$

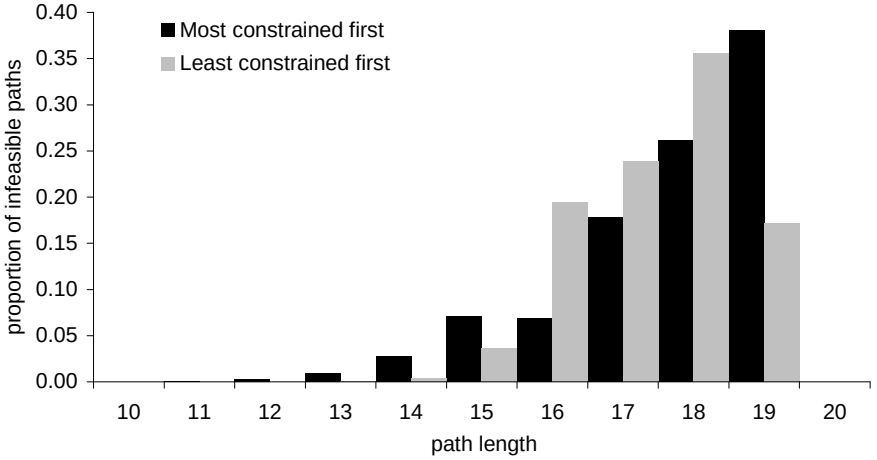
where $nocap = 0.1$ is used to avoid division by zero (capacities are integral values) and favour tasks that have a restricted choice of available agents. Although both measures are similar, empirical investigation reveals they produce differing results.

Regardless of the measure used, assigning tasks in non-increasing order of constrainedness would be expected to lead to infeasible solutions being found earlier in solution construction than would otherwise be the case, but also to reduce the number of these infeasible solutions. Figure 3.7 compares ordering items by non-increasing and non-decreasing order of constrainedness with respect to the number of infeasible solutions and the distribution of their lengths for the first instance in the `gap2` instance. Only solutions of length 20 are feasible.

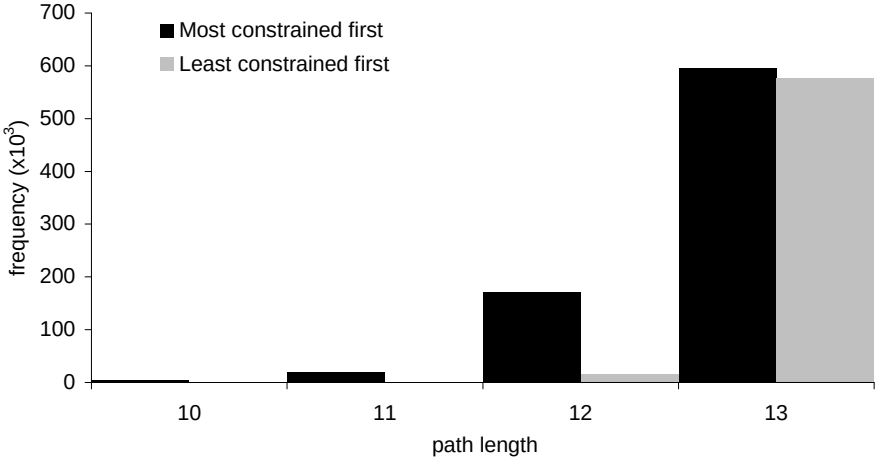
Sampling of the construction trees produced by 3000 randomly generated static assign-



(a) Frequency of path lengths



(b) Frequency of path lengths as a proportion of infeasible solutions



(c) Frequency of path lengths between 10 and 13

Figure 3.7: Distribution of path lengths for the first `gap2` instance under different assignment orders.

ment orders for instances from the gap1 (5 agents, 15 tasks) and 1000 for instances from the gap2 (5 agents, 20 tasks) problem sets (Beasley, 2005) reveals that very few static assignment orders can produce a relatively high probability of finding feasible solutions. Figure 3.5 shows the distribution of feasible solution probabilities achieved by this sample for the five instances in the gap2 problem set. Comparison with the construction tree produced by a static assignment order in which tasks appear in non-increasing order of constrainedness indicates that it can produce a higher probability of finding feasible solutions than 90% of alternative static orders for gap1 instances (although on one instance it was better than only 24% of sampled orders) and 80% of some gap2 instances.

A number of other assignment order heuristics (referred to simply as assignment orders hereafter) were examined with regards to the probability of reaching a feasible solution they gave on the gap1 and gap2 instances. A brief description of each appears below.

Static, fixed order (SFO). Tasks are assigned in the same order they appear in the problem instance description.

Static, most constrained first (SMC). Tasks are sorted by non-increasing order of constrainedness. All solutions are constructed using this order.

Static, least constrained first (SLC). Tasks are sorted by non-decreasing order of constrainedness. All solutions are constructed using this order.

Dynamic, most constrained first (DMC). Currently most constrained task is next to be assigned.

Dynamic candidate set (DCS). A candidate set is formed from the most constrained task for each agent, i.e., each agent provides one task for the set. The next task to be assigned is chosen with a uniform random probability over the candidate set.

Dynamic, probabilistic (static measure) (DPS). The next task to be assigned is selected probabilistically in proportion to an *a priori* measure of its constrainedness.

Dynamic, probabilistic (dynamic measure) (DPD). The next task is selected probabilisitcally in proportion to its current level of constrainedness.

Tables 3.1 and 3.2 show the performance of each assignment order when used with the 10 instances from the gap1 and gap2 problem sets (each contains five instances). The tables report the number of infeasible paths in the construction trees (denoted by **infeas.**), the probability of producing a feasible solution using an undirected search (denoted by P(**feas.**)), and the proportion of static assignment orders this probability is greater

than (denoted by **%ile**). Results for the `gap1` instances were produced by complete enumeration of the construction trees for each assignment order–instance combination. Non-deterministic dynamic assignment orders (i.e., DCS, DPS and DPD) were run across 10 random seeds, with the mean probability of producing a feasible solution taken. The number of infeasible solution paths for the `gap2` instances were produced by complete enumeration of the construction trees, while the results for P(feas.) were produced by examining 30000 solutions produced by $\text{ACO}_{\text{undir}}$.[5]

Both constrainedness measures described by Equations 3.5 and 3.6 were investigated with the SMC, SLC, DMC and DCS assignment orders, with each assignment order annotated by 'a' when using the absolute measure of constrainedness defined by Equation 3.5 and 'p' when using the proportional measure of constrainedness defined by Equation 3.6. DPS and DPD were tested with the proportional measure of constrainedness only. Results for DCSa and DCSp were equivalent on the instances studied and so only one set of data is given. Although not statistically significantly different, SMCa and SMCp produce some differing results and so are both reported.

Given the variability in P(feasible) results across instances, the results of pairs of assignment orders were compared using their percentile ranks (i.e, the percentage of random static assignment orders they outperform). Due to the small sample (five instances from each problem set) the Mann-Whitney test was used to determine if the observed differences were statistically significant. Tables 3.3 and 3.4 show the results of the comparisons for the `gap1` and `gap2` problem sets respectively. The tables are read as follows. To compare assignment orders A and B, locate A in the left column and then locate the relationship indicator, such as $<$, in the column under B, which gives $A < B$. Such an entry means that A gives a lower probability of producing feasible solutions than B. If the result is statistically significant then the significance level is shown below the relationship indicator. If the probability that the observed difference is due to chance is above 85%, the $=$ symbol is used instead of $<$ or $>$.

In general, all heuristic assignment orders examined, with the exception of DCS, achieve a relatively high probability of reaching a feasible solution given what problem constraints allow. The chief exception is the third `gap1` instance, in which only three assignment orders were above the 50^{th} percentile. Nevertheless, the results suggest that relatively simple assignment orders can be effective, at least on small GAP instances.

The results may be used to produce a coarse ranking of the approaches. In the following $A \prec B$ is used to indicate that there is strong statistical evidence that B outperforms A, $A \preceq B$ indicates that B generally outperforms A, but the result is not statistically

[5]As the probability of any one sequence is extremely small in these instances, the software used for complete enumeration of construction trees was unable to accurately record the aggregate probability of solutions. Consequently, an actual implementation of $\text{ACO}_{\text{undir}}$ was used to estimate these probabilities.

Table 3.1: Feasible probability achieved by different assignment orders for `gap1` GAP instances. Bold entries denote the best result found for that instance.

		Assignment order								
Instance		SFO	SMCp	SMCa	SLC	DMCp	DMCa	DCS	DPS	DPD
1	infeas. ($\times 10^3$)	873.3	109.1	109.1	2437.7	272.9	**107**	210.5	138.3	255.7
	P(feas.) (%)	0.27	1.40	1.39	0.25	1.09	1.45	1.04	**1.53**	1.28
	%ile	7.9	89.3	89	5.8	75.4	91	71.8	**93.3**	85
2	infeas. ($\times 10^3$)	583.2	**56.6**	60.5	681.1	118.8	60.5	150.7	73.4	135.1
	P(feas.) (%)	0.01	**0.45**	0.41	0.03	0.26	0.41	0.08	0.42	0.26
	%ile	2	**99.4**	99.1	16.5	96.3	99.1	66.5	99.1	96.3
3	infeas. ($\times 10^3$)	476.2	169.2	**146**	1360.2	297	**146**	255.6	185.4	293.7
	P(feas.) (%)	1.5	1	**1.6**	0.9	1.2	**1.6**	1.2	1.5	1.4
	%ile	52.2	24.5	**56.8**	20.3	36.2	**56.8**	34	51.5	45.9
4	infeas. ($\times 10^3$)	489.8	**48**	**48**	1319.3	127.4	48.1	87.3	59.5	122.3
	P(feas.) (%)	0.29	**4.40**	**4.40**	0.10	3.13	4.32	2.61	3.83	3.03
	%ile	18	**99.8**	**99.8**	0.4	99.1	99.8	98.3	99.6	99
5	infeas. ($\times 10^3$)	906	**100.2**	110.5	1688.5	280.4	104.2	316.9	138.3	298.7
	P(feas.) (%)	0.91	10.67	10.30	0.47	6.77	**11.11**	3.03	8.04	5.65
	%ile	25.5	**100**	**100**	10.5	99.4	**100**	80.5	99.8	98.4

Table 3.2: Feasible probability achieved by different assignment orders for `gap2` GAP instances. Bold entries denote the best result found for that instance.

		Assignment order								
Instance		SFO	SMCp	SMCa	SLC	DMCp	DMCa	DCS	DPS	DPD
1	infeas. ($\times 10^6$)	437.4	65.7	**64.4**	1951.7	136.2	65.8	104	76.9	144.3
	P(feas.) (%)	0.50	1.22	1.30	0.13	0.86	1.08	0.88	**1.37**	0.96
	%ile	48.2	90.9	92.9	2.5	80.7	88.4	81.5	**93.7**	85.2
2	infeas. ($\times 10^6$)	488	89.8	**87.8**	1819.4	195.3	89	195.3	108	205.2
	P(feas.) (%)	0.71	**1.21**	1.20	0.21	0.81	1.17	0.67	1.17	0.82
	%ile	58.6	**89.2**	**89.2**	3.7	67	88.1	55.4	88	69
3	infeas. ($\times 10^6$)	536.5	274.7	314.9	2016.6	596.2	**260.1**	556.7	331.7	604.9
	P(feas.) (%)	1.61	2.98	2.98	2.40	2.16	**3.40**	1.49	3.24	2.52
	%ile	13.6	67.4	67.6	45.1	35.8	**79.8**	9.2	75.2	50.4
4	infeas. ($\times 10^6$)	807.7	**112.5**	**112.5**	1885.9	297.4	114.5	317.7	150.6	305.2
	P(feas.) (%)	1.22	**3.38**	3.27	0.61	2.27	3.33	1.23	2.97	2.39
	%ile	38	**94**	93.3	7	78.1	93.6	38	90.3	80
5	infeas. ($\times 10^6$)	386.8	**191.4**	194.9	2436.9	474.5	195.4	374	238.6	478
	P(feas.) (%)	**4.92**	4.26	4.04	1.73	3.02	3.99	2.86	4.15	3.32
	%ile	**90.5**	82.9	78.8	11.1	50.7	78	44.6	81.6	59.5

significant, and $A = B$ indicates that A and B perform equivalently. Based solely on results for the `gap1` instances,

$$\text{SLC} \preceq \text{SFO} \prec \text{DCS} \preceq \text{DPD} \preceq \text{DMCp} \preceq \text{DPS} \preceq \text{SMCp} = \text{SMCa} \preceq \text{DMCa},$$

while based solely on results for the `gap2` instances,

$$\text{SLC} \prec \text{DCS} \preceq \text{SFO} \preceq \text{DMCp} \preceq \text{DPD} \prec \text{DMCa} \preceq \text{SMCa} \preceq \text{DPS} \preceq \text{SMCp}.$$

If results for the two instances are considered together,

$$\text{SLC} \prec \text{SFO} \prec \text{DCS} \preceq \text{DMCp} \preceq \text{DPD} \preceq \text{SMCp} = \text{SMCa} = \text{DMCa} = \text{DPS}.$$

Considering the results for both problem sets, heuristically determining a static assignment order appears to be as effective as more complicated dynamic heuristics such as DPS and DMC, at least on small GAP instances. Given that dynamic assignment orders require greater computational resources, static assignment orders may offer advantages over dynamic heuristics. Chapter 7 reports on the performance of ACO using a range of assignment orders on larger GAP instances.

3.4 Examples of Bias in Constructive Algorithms

This section describes some common COPs to which constructive approaches such as ACO have been applied and considers what biases they possess. Details of the standard approach to applying ACO to those problems not discussed previously, and descriptions of the biases they exhibit as a consequence, are as follows:

MCP Solutions to this problem may be built by successively including nodes in the clique under construction (e.g., Fenet and Solnon, 2003). As nodes may be added in any order and solutions may be different lengths, this problem has a representation bias. In terms of construction bias, shorter sequences will have a higher probability of being produced than longer ones, while the branching on some paths will be non-uniform given the existence of infeasible combinations of nodes.

GCP This problem may be posed in two ways, with the objective of minimising either the total number of colours used or the number of colour conflicts for a given number of colours. In the former, all solutions generated are feasible, while in the latter, technically infeasible solutions do not halt solution construction (i.e., they are feasible in terms of what the algorithm may produce). Consequently, they are both free from a construction bias due to infeasible or variable length solutions. However, as alternative representations of each solution may be obtained by cycling the colours used

Table 3.3: Pairwise comparison of assignment orders for `gap1` GAP instances. Direction of difference between percentile ranks of pairs of assignment orders are shown. If the difference is statistically significant (based on a Mann-Whitney test) then the significance level is shown below the direction indicator.

	SMCp	SMCa	SLC	DMCp	DMCa	DCS	DPS	DPD
SFO	< (α =5%)	< (α =1%)	>	< (α =1%)	< (α =1%)	< (α =1%)	< (α =1%)	< (α =1%)
SMCp		=	> (α =1%)	>	<	>	>	>
SMCa			> (α =1%)	>	<	> (α =10%)	>	>
SLC				< (α =1%)	< (α =1%)	< (α =1%)	< (α =1%)	< (α =1%)
DMCp					<	>	<	>
DMCa						> (α =10%)	>	>
DCS							< (α =10%)	<
DPS								>

Table 3.4: Pairwise comparison of assignment orders for `gap2` GAP instances. Direction of difference between percentile ranks of pairs of assignment orders are shown. If the difference is statistically significant (based on a Mann-Whitney test) then the significance level is shown below the direction indicator.

	SMCp	SMCa	SLC	DMCp	DMCa	DCS	DPS	DPD
SFO	< (α =5%)	< (α =5%)	> (α =5%)	<	< (α =5%)	<	< (α =5%)	<
SMCp		=	> (α =1%)	> (α =5%)	>	> (α =1%)	>	> (α =5%)
SMCa			> (α =1%)	> (α =5%)	>	> (α =5%)	>	> (α =10%)
SLC				< (α =1%)	< (α =1%)	< (α =5%)	< (α =1%)	< (α =1%)
DMCp					< (α =5%)	>	< (α =5%)	<
DMCa						> (α =5%)	>	> (α =5%)
DCS							< (α =1%)	< (α =10%)
DPS								>

between different groups of like-coloured nodes, if a solution requires more colours then it will also have more representations. Consequently, the first formulation of the problem has a representation bias, while the second does not. In both cases the order in which nodes are assigned colours will determine how soon a partial solution either becomes infeasible or requires a new colour to be used.

SCP This problem consists of selecting a subset of items, each of which is said to cover certain constraints, such that all constraints are covered (Fiorenzo Catalamo and Malucelli, 2001). As the order in which items are selected is unimportant, and varying numbers of items are required to produce a feasible solution, this problem has both a representation bias and a construction bias due to variable length solutions. As the algorithm stops as soon as a cover is generated, certain combinations of items can never be produced and can be considered infeasible, which also contributes to the construction bias.

SPP This problem consists of selecting a subset of items, each of which is said to cover certain constraints, such that the items selected partition the constraints (Maniezzo and Milandri, 2002). It is subject to the same causes of bias as the SCP, with the added construction bias that many shorter paths correspond to unavoidable infeasible solutions.

Table 3.5 summarises the sources of bias in constructive algorithms for the problems described in this chapter. Alternative constructive algorithms for the GSP and assignment problems are presented below to illustrate how changing the algorithm can affect the biases exhibited.

3.4.1 Solving the JSP, GSP or OSP as Assignment Problems

Scheduling problems such as the JSP, GSP and OSP may be transformed into assignment problems either by directly assigning processing times to the operations, or by assigning directions to the undirected arcs in their disjunctive graph representation. The latter is discussed here. Taking this approach, $\mathfrak{C} = \mathfrak{C}_{it} \times \mathfrak{C}_{res}$ where $\mathfrak{C}_{it} = \{(i,j) \in \mathcal{O} \times \mathcal{O} \mid G(i) = G(j) \vee M(i) = M(j)\}$, $\mathcal{O}$ is the set of operations to be scheduled, $G(i)$ is the group i belongs to, $M(i)$ is the machine on which i must be processed, and $\mathfrak{C}_{res} = \{\text{true}, \text{false}\}$ where the assignment $((i,j), \text{true})$ means that operation i is to be scheduled before operation j, while $((i,j), \text{false})$ indicates the reverse.

Constructing solutions in this way, each solution has only one representation, thus eliminating the representation bias present when solutions are constructed as permutations of the operations. However, there is still a construction bias as some combinations of relative orderings create contradictions which cannot therefore represent feasible solutions

Table 3.5: Examples of bias in constructive algorithms for different COPs. Where an asterisk (*) appears, the variable length solutions for those problems are caused at least in part by unavoidable infeasible solutions.

		Construction bias influences		
Problem	Representation bias	Infeasible solutions	Variable length solutions	Assignment order changeable
TSP	—	—	—	n/a
MKP	✓	✓	✓	n/a
MCP	✓	✓	✓	n/a
QAP	—	—	—	✓[a]
JSP, GSP	✓	✓	—	n/a
OSP	✓	—	—	n/a
GAP	—	✓	✓*	✓
GCP[b]	✓	—	—	✓
GCP[c]	—	—	—	✓
SCP	✓	✓	✓	n/a
SPP	✓	✓	✓*	n/a

[a]Alters solutions' positions in construction tree, but does not create a bias

[b]When aim is to minimise number of colours used

[c]When aim is to minimise number of colour conflicts

(although there are no infeasible partial solutions). This is true when applying this approach to the JSP, GSP *and* OSP. As the problem is formulated as an assignment problem, different assignment orders may be used. Figure 3.8 shows the construction tree resulting from the application of this approach to the `jsp2-2` instance described earlier using a randomised assignment order.

Although this alternative construction approach to these problems appears to be less biased than the list scheduler algorithm typically used, it has the disadvantage that candidate directions must be eliminated if they would create a cycle, which is more time consuming than maintaining the list of available operations in the list scheduler algorithm.

3.4.2 Solving Assignment Problems with No Assignment Order

Most ACO algorithms for assignment problems separate the choice of which item to assign next from the choice of which resource to assign. That is, at each step, they select an item either heuristically or randomly, and then choose a resource to assign to that item. Roli et al.'s (2001) ACO algorithm for constraint satisfaction takes a different approach, in which solution components are complete assignments. At each step, any individual assignment of an item to a resource may be chosen. This same approach is used in a GRASP for constraint satisfaction by Resende, Pitsoulis and Pardalos (1997).

Using this alternative construction approach, the problem of selecting an assignment

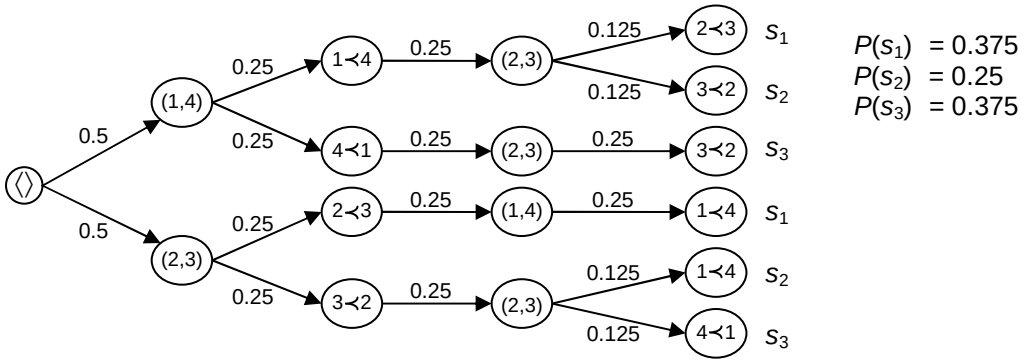


Figure 3.8: Construction tree for `jsp2-2` instance when solutions are produced by assigning an order of precedence between pairs of related operations. $i \prec j$ means that the decision has been made that operation i should precede related operation j in the solution. Arcs are labelled with the probability of their being traversed if using $\text{ACO}_{\text{undir}}$. End points are labelled with the solution represented by that path. Aggregate solution probabilities appear on the right.

order is subsumed by the problem of selecting a collection of assignments. The construction tree induced by this approach is isomorphic with the construction tree produced by a dynamic, randomised assignment order (if decisions regarding which item to assign at each step are considered to be interleaved between decisions about actual assignments in the tree). Consequently, it potentially gives access to any particular static or dynamic assignment order. Therefore, observable biases will be similar to those seen using other dynamic assignment orders in a typical construction algorithm for these problems. In problems where assignment order can have a significant influence on performance, such as in the GAP, it may become necessary to incorporate features of any assignment order into the rule for selecting amongst competing assignments.

3.5 Bias Effects and Problem Size

As problem instance size grows it becomes impossible to perform a complete exploration of the construction tree, and so the impact of construction and representation biases cannot be analysed precisely. While these biases are still present, as the underlying mechanisms do not change with problem size, any constructive search algorithm can at best produce a sample of the many feasible solutions to such instances, making it difficult to observe their effects on larger instances. Analysis of small instances shows that as problem size grows, the relative difference in probability between individual sequences (and solutions) becomes very large. However, given that the number of solutions grows exponentially with instance size in COPs, even if a single solution's probability is relatively high compared to another solution's, its probability is still extremely low overall. Table 3.6 shows the minimum

Table 3.6: Bias effects and instance size.

Instance	Size	Feasible solutions	Solution probability min	Solution probability max	Ratio (rounded)
MKP					
mknap1-6item	6 items	8	0.0833	0.3333	4
mknap1-10item	10 items	19	0.0094	0.1564	17
mknap1-15item	15 items	708	0.0002	0.0376	175
GAP					
gap1, instance 1[a]	15 tasks, 5 agents	55996	1.9×10^{-9}	9.3×10^{-4}	480001
gap1, instance 1[b]	15 tasks, 5 agents	55996	5.6×10^{-10}	1.3×10^{-4}	225000
gap2, instance 1[a]	20 tasks, 5 agents	75.1×10^{6}	2.7×10^{-14}	6.9×10^{-5}	2.5×10^{9}
gap2, instance 1[b]	20 tasks, 5 agents	75.1×10^{6}	3.6×10^{-14}	1.6×10^{-6}	42.9×10^{6}
GSP					
jsp2_2	4 operations	3	0.25	0.5	2
jsp3_3[c]	9 operations	64	0.001	0.099	96

[a]Using best assignment order.

[b]Using worst assignment order.

[c]Details of this instance given in Section 5.2.

and maximum solution probabilities for small MKP, GAP and JSP instances. Instances from the `mknap1` problem set are labelled in the same manner as the `mknap1-15item` and `mknap1-50item` instances.

Although the effects of solution biases cannot be observed on larger instances due to the large number of solutions, the mechanisms that introduce these biases, in particular those that cause construction bias, also introduce a bias to any pheromone representation used. This issue is discussed in Chapter 5.

3.6 Deliberately Introduced Sources of Bias

The analyses in this chapter have all assumed the use of a constructive algorithm ($\text{ACO}_{\text{undir}}$) which makes decisions using a uniform probability over the available components at each step. This assumption was a necessary simplification in order to study the underlying sources of biases that may affect constructive algorithms. These biases determine the baseline probability of reaching particular sequences and hence, solutions. However, actual constructive optimisation algorithms are not undirected and typically employ a range of dynamic techniques to bias their searches towards good solutions. All of these deliberately introduced sources of bias adjust sequences' and solutions' respective probabilities away from the baselines established by the underlying biases in the constructive approach used for a problem. A number of these techniques are discussed here.

The simplest deliberately introduced source of bias is the use of heuristic information to

indicate which solution components appear good, at least in the short term. Certainly this is an important component of most ACO implementations. Heuristic information may be static (such as taking the inverse of the distance between the current city and a candidate city in the TSP), or dynamic (as in some ACO applications for the VRP, e.g., Doerner et al. (2002)). In a probabilistic constructive algorithm such as ACO, heuristic information adjusts the relative probabilities of alternative choices at each node in the construction tree. Without specialised insight, complete enumeration of the construction tree for a problem is required in order to understand the combined effects of the use of heuristic information and any underlying biases. An approximation of these effects may also be obtained by comparing the solutions found by $\text{ACO}_{\text{undir}}$ and an $\text{ACO}_{\text{undir}}$ algorithm that uses heuristic information to adjust the probability of selecting each solution component (denoted by ACO_{heur}).

A related source of deliberate bias comes from the use of candidate sets. Typically, the set of candidate components is small and consists of those components that appear good based on some heuristic measure. Thus candidate sets can be used to both increase algorithm speed (by decreasing the number of solution components considered at each step) and improve the quality of solutions produced by introducing a strong bias in favour of heuristically good components. In essence, they are a more extreme form of the use of heuristic information, in that the probability of a number of components is reduced to zero. Static candidate sets (i.e., those based on static heuristic information) have been used in a number of ACO algorithms for the TSP to enable their application to large instances (e.g., Dorigo and Gambardella, 1997a; Stützle and Hoos, 1998). As with heuristic information, analysis of either the full construction tree or the results of $\text{ACO}_{\text{undir}}$ and an $\text{ACO}_{\text{undir}}$ or ACO_{heur} algorithm that uses candidate sets is required to understand the bias they introduce. ACO algorithms for the TSP and a car sequencing problem developed by Randall and Montgomery (2002) make use of dynamic candidate sets, which regenerate the candidate sets periodically using both pheromone and heuristic information. Such candidate sets are part of a complex feedback system, which makes analysis of any bias they introduce extremely difficult. However, ACO algorithms that have made use of either static or dynamic candidate sets have achieved improvements in computation time and solution quality.

Intensification and diversification techniques are useful additions to most metaheuristics (Glover and Laguna, 1997). Intensification involves focusing a search on solutions that contain particular features (typically those that appear to be part of good solutions), while diversification forces the search away from features of solutions that have been seen before. The two techniques would appear to be mutually exclusive, although this is not the case as both can operate on different parts of a single solution, locking in some parts while forc-

ing other parts to use previously unused features. Glover and Laguna state that the two techniques should often be used at the same time. In ACO algorithms that use these techniques, intensification and diversification typically occur at the level of individual solution components, although other approaches have also been used. Intensification and diversification techniques have been used in ACO algorithms by Blum (2002a), Gambardella, Taillard and Dorigo (1999), Randall (2003a), Randall and Tonkes (2002) and T'kindt et al. (2002). In terms of the possible effects of these techniques, intensification will generally increase certain components' probabilities, while diversification may either decrease certain components' probabilities or make all components' probabilities more uniform, depending on the exact technique used. This latter kind of diversification mechanism has been used by Gambardella, Taillard and Dorigo (1999) and Stützle and Hoos (1998).

It is common practice in many constructive metaheuristics to apply a local search procedure to the solutions produced by construction (Dorigo and Stützle, 2004; Feo and Resende, 1995). The use of such procedures, which are typically very greedy, introduces solution *wells* into which a number of solutions will fall. Accordingly, the probability of reaching a solution s in a well is proportional to the collective probabilities of all those solutions which the local search procedure transforms into s. Although such techniques introduce a strong bias, evidence from the literature clearly indicates they help to produce very good solutions, and so may serve to counteract an otherwise unfavourable bias. However, care must be taken in their design to ensure that they give adequate access to different areas of the search space. A poorly designed local search procedure may make it difficult to reach the optimal solution and may even, given a starting solution close to the optimum, move the search further away from the optimum (Reeves, 1999).

A largely ACO specific trait is the use of an artificial pheromone to bias the search towards those solution components that have appeared in good solutions in the past. This feature is central to the ACO algorithm and as such is the subject of the remaining chapters of this thesis.

3.7 Summary

Constructive metaheuristics explore a tree of constructive decisions as they build solutions. The nature of the problem being solved and the constructive algorithm used can combine to introduce a bias into the search. A *representation* bias occurs when solutions are represented by differing numbers of sequences in the construction tree, while a *construction* bias is present when problem constraints lead to a restriction on the degree of branching at some paths, thereby increasing the probability of sequences on those paths. Collectively these biases are referred to as constructed solution biases.

A construction bias can also be produced in problems where solutions are of variable length, such as in subset problems, where short solutions have an increased probability of being found. In highly constrained problems, such as the GAP, short solutions may exist only as infeasible dead-ends in the construction tree. This can be problematic as these infeasible solutions consequently have an elevated probability.

When building solutions to assignment problems such as the QAP and GAP, altering the order in which items are assigned alters the topology of the construction tree and changes which solutions are near each other. In assignment problems with infeasible solutions, selecting a good assignment order can reduce the probability of producing an infeasible solution. Relatively simple, commonly used heuristics for selecting an appropriate assignment order for the GAP appear to work well with some problem instances, but are occasionally worse than the majority of randomly chosen static assignment orders.

Although the analyses in this chapter assumed an undirected constructive algorithm, actual metaheuristics are not undirected, and employ a range of techniques to deliberately bias their searches. These deliberate biases may be either to guide the search towards solutions (or parts of solutions) that appear promising or to diversify the search and possibly escape from local optima. In a constructive setting, representation and construction biases will set a baseline probability for each solution. Deliberately introduced biases do not replace these probabilities, but alter them. Further investigation is required to understand how these deliberately introduced biases interact with underlying solution biases.

Nevertheless, understanding the nature of the biases that may exist in constructive algorithms can assist in the development of improved ACO algorithms. Pheromone information, the distinguishing feature of the ACO metaheuristic, is an important deliberately introduced source of bias that interacts with these underlying constructed solution biases. The nature of the interaction may make some pheromone representations more effective than others, especially in those cases where there exists a consistent, predictable pattern in the interaction. These issues are discussed in Chapter 5. However, before the combined effects of bias and pheromone can be described it is necessary to study pheromone representations in more detail, which is the topic of the next chapter.

Chapter 4

Pheromone Representations

The use of an artificial pheromone to learn the value of solution components is the defining feature of ACO; without it an ACO algorithm merely becomes a probabilistic greedy constructive heuristic. The pheromone representation used by a particular ACO algorithm is a model of aspects of solutions to the problem being solved. ACO thus belongs to the class of model-based search (MBS) algorithms (Zlochin and Dorigo, 2002). To date, no thorough review of the range of pheromone representations used has been conducted and no consistent language has appeared in the ACO literature to describe formally what different pheromone representations model.

This chapter considers the different features of solutions that pheromone may be used to model and how particular problems and constructive algorithms for those problems restrict the set of suitable pheromone representations (or models). While some of the different approaches to updating pheromone values are discussed, the main purpose of the chapter is to describe how pheromone may be used to model solutions. Section 4.1 introduces a notation for describing pheromone representations in terms of the solution components used to build solutions and discusses the two main approaches to deriving a pheromone representation for a given problem and constructive algorithm. The notation is tailored to suit the majority of pheromone representations that have been used in the literature, but some atypical pheromone representations have been used for which it is not well suited. These are discussed in Section 4.1.2. Section 4.2 discusses issues that are peculiar to the use of higher order pheromone representations—models of combinations of solution components.

4.1 Formalisation of Pheromone Representations

A pheromone representation is a model used to guide ants as they construct solutions, and so maps certain identifiable features of solutions to pheromone values. Hence, the nature

of a pheromone representation is determined by the features it models. Typically, these features correspond to the decision variable–value pairs in the problem being solved. For a given combinatorial optimisation problem often several alternative sets of decision variables may be defined and consequently different pheromone representations are also possible. As an extension to the definition of a solution component given in Chapter 2, it is useful to introduce the concept of a *solution characteristic*, denoted by c, which is equivalent to the binding of a value to one of the decision variables for a problem. Modelling a set of solution characteristics implicitly defines a set of binary decision variables, as each solution may or may not exhibit each particular solution characteristic.

An individual characteristic may correspond to the presence of a particular solution component or to some broader feature of solutions produced by a number of solution components or other entities in the problem. For instance, in the MKP, the solution characteristics typically modelled represent the presence of a particular component (i.e., item) in the solution. An example of the second case is the pheromone representation used for the TSP, in which solution characteristics correspond to pairs of successive components (i.e., edges of the graph of connected cities). A set of solution characteristics may be denoted by combinations of the entities to which the characteristics relate. These can include the set of components $\mathfrak{C}$ from which solutions are built or, in assignment problems, the sets of items $\mathfrak{C}_{it}$ or resources $\mathfrak{C}_{res}$ (where in many cases sequences are constructed from elements of one set only, but which implicitly represent assignments). It may also be generalised to make use of any number of different types of entity $\mathfrak{C}_1, \ldots, \mathfrak{C}_n$ in a problem. Some pheromone representations also model the absolute position of a component in a sequence, the set of positions being denoted by P. Thus, the solution characteristics modelled for the MKP may be denoted simply by $\mathfrak{C}$, while those for the TSP may be denoted by $\mathfrak{C} \times \mathfrak{C}$. The set of assignments (each of which is a solution characteristic) in an assignment type problem such as the QAP may be denoted by $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$, where $\mathfrak{C}_{it}$ is the set of facilities and $\mathfrak{C}_{res}$ the set of locations.

Denote an arbitrary, unspecified set of solution characteristics by C and a subset of solution characteristics corresponding to a solution s as $C_s \subset C$. Strictly speaking, a pheromone representation is a mapping from solution characteristics C to pheromone values and so should be denoted by $C \mapsto T$. Except in a small number of cases, which are clearly identified, pheromone representations associate a single pheromone value to each solution characteristic, and so $C \mapsto T$ can be abbreviated to C.

The notation is not intended to be—indeed, will rarely be—a precise mathematical definition of the solution characteristics, and hence pheromone values, modelled by a pheromone representation in an actual application. However, it does not preclude such definitions and may form an intermediate step in their specification. For instance, in the ATSP, a

precise definition of the solution characteristics modelled is $C = \{(\mathfrak{c}_i, \mathfrak{c}_j) \in \mathfrak{C} \times \mathfrak{C} \mid \mathfrak{c}_i \neq \mathfrak{c}_j\}$. Hence, $C \subset \mathfrak{C} \times \mathfrak{C}$, which is what the notation defines for this problem. Similarly, the symmetric TSP has the same definition for C with the additional constraint that $\tau(\mathfrak{c}_i, \mathfrak{c}_j) = \tau(\mathfrak{c}_j, \mathfrak{c}_i)$. The notation introduced here thus provides for a typically short description of the set of solution characteristics modelled by a pheromone representation, while additionally corresponding directly to computer implementations of such representations, which use arrays and matrices and hence include a number of unused pheromone values.

Where the addition of a single solution component to a partial solution corresponds with the addition of a single solution characteristic, the pheromone representation in question is said to be *first order* (Blum, 2004; Blum and Sampels, 2002a). In such cases, a single pheromone value from the representation is used to influence an ant's decision regarding a single solution component. *Higher order* representations involve subsets of solution characteristics influencing each decision. For instance, a second order pheromone representation may indicate the utility of having a pair of solution components in the same solution. Thus, higher order pheromone representations implicitly define two sets of decision variables for a problem: one set relates to decisions to include individual solution components based on the current state of a partial solution, while the other set relates to higher order decisions, i.e., whether a single solution should exhibit two or more solution characteristics at the same time. Such pheromone representations may emerge in two ways. Given an existing first order pheromone C, the n^{th} order pheromone representation may be obtained by transforming it into C^n, where the pheromone associated with the n-tuple $(c_i, \ldots, c_k) \in C^n$ represents the learned utility of having all n characteristics $c_i, \ldots, c_k$ in the same solution. For instance, in a subset problem where $C = \mathfrak{C}$, the n^{th} order pheromone $C^n = \mathfrak{C}^n$ represents the utility of different combinations of n components being part of the same solution. Alternatively, when the solution characteristics modelled relate many parts of the solution to each other, a higher order pheromone forms naturally as a consequence of having to combine information from each of those relationships. In this case, it is often impossible to use the underlying first order representation independently, unlike the former situation. In both cases, the resulting representation is denoted by $\mathfrak{S}^p \times C; C^n$, where $\mathfrak{S}^p$ represents the set of all partial solution sequences. The first part describes the *observed* pheromone representation, where the pheromone associated with adding a solution characteristic from C is contingent on a partial solution from the set $\mathfrak{S}^p$. The observed pheromone representation provides a single pheromone value for each candidate solution component and hence is equivalent to a first order representation. Therefore, ants make decisions based on the values in this observed pheromone representation. The second part describes the underlying pheromone representation C^n from which pheromone

values are aggregated (various approaches are considered in Section 4.2) to produce the observed pheromone representation. An example appears below.

The strict definition of the n^{th} order pheromone representation generated from C includes only higher order solution characteristics (and hence pheromone values) for n-tuples consisting of n *distinct* elements from C. However, the simpler notation C^n is used.[1] The structure of a typical n^{th} order pheromone is described formally in Definition 3.

Definition 3 The n^{th} order pheromone representation derived from the first order pheromone representation C is $\{(c_1, \ldots, c_n) \mid c_1, \ldots, c_n \in C,\ c_1 \prec \ldots \prec c_n\} \subset C^n$ where $c_i \in C$ is a solution characteristic from C and $\prec$ is an arbitrary fixed order imposed on the elements of C. This is denoted by $\mathfrak{S}^p \times C; C^n$. □

An example of a naturally occurring second order pheromone is that used for the GCP (Costa and Hertz, 1997). Solutions are produced by the assignment of colours to nodes, which suggests solution characteristics (and hence, decision variables) from $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ where $\mathfrak{C}_{it}$ is the set of nodes and $\mathfrak{C}_{res}$ the set of colours. However, Costa and Hertz use a pheromone representation of the form $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$, where $\mathfrak{C}_{it} \times \mathfrak{C}_{it}$ represents a pair of nodes being assigned the same colour. Hence, the pheromone representation naturally relates a number of solution components. Although Definition 3 cannot be applied directly to this second order pheromone, the standard C–C^2 relationship can still be seen by considering a typical second order pheromone where solution characteristics are pairs of node–colour assignments, $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; (\mathfrak{C}_{it} \times \mathfrak{C}_{res})^2$. That is, where solution characteristics correspond to the solution components used to build solutions. Given that the actual colours assigned to nodes do not affect the cost of solutions—only what nodes are in the same colour group—all references to actual colours in the underlying $(\mathfrak{C}_{it} \times \mathfrak{C}_{res})^2$ pheromone may be removed, producing the simpler $\mathfrak{C}_{it} \times \mathfrak{C}_{it}$ pheromone. This second order pheromone is referred to as a *grouping pheromone* hereafter.

Table 4.1 shows the notation applied to a number of commonly used pheromone representations. Where the solution characteristics described by C are potentially ambiguous it may be necessary to specify their proper interpretation. The third and fourth representations in Table 4.1 are both described as $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$, where $(i, j) \in \mathfrak{C} \times \mathfrak{C}$ may represent i and j being copresent in a solution or that i appears before a related item j in the solution. These can be specified more clearly by denoting the representations as $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) respectively.

[1] This corresponds to the use of matrices to implement such representations given their fast access properties not possessed by sparse representations.

Table 4.1: Common pheromone representations. Example problems represent a sample of how representations have been used in the literature. In some cases different pheromones have been used with the same problem.

Pheromone	Pheromone associated with...	Example problems
$\mathfrak{C}$	items present in solution	MKP, SCP
$\mathfrak{C} \times \mathfrak{C}$	one item succeeding another	JSP, SMTTP, TSP
$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)	pairs of items present in solution	MCP
$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding)	one item preceding a collection of others	JSP
$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$	pairs of items in same/different group	GCP
$\mathfrak{C} \times P$	position of item in solution	JSP, SMTTP
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	assignment of resource in $\mathfrak{C}_{res}$ to item in $\mathfrak{C}_{it}$	GAP, QAP

4.1.1 Representation-Oriented and Identity-Oriented Pheromones

Two approaches may be taken to derive a pheromone representation from the problem model used by ants to build solutions: representation-oriented and identity-oriented. The former produces pheromones that reflect some aspect of *how* solutions are represented (i.e., the arrangement of solution components in a solution sequence), while the latter results in pheromones that describe *which* solutions are represented. These are specified more formally in Definitions 4 and 5.

Definition 4 A pheromone representation C is said to be *representation-oriented* if, for each sequence $\mathfrak{s}$, there is a unique set of solution characteristics drawn from C that describes only $\mathfrak{s}$. □

Definition 5 A pheromone representation C is said to be *identity-oriented* if, for each solution s, there is a unique set of solution characteristics drawn from C that describes s.□

The arrangement of solution components may identify solutions in some problems, but in many cases only indirectly indicates the identity of the solutions represented. For example, representing solutions to a subset problem as a linear list of items, a $\mathfrak{C} \times P$ pheromone could be used to learn where to place items in the sequence. These solution characteristics imply decision variables of the form x_i representing the position of item i in the solution. This indirectly indicates which items should be chosen. Using a $\mathfrak{C}$ pheromone for a subset problem represents the identity of solutions quite separately from how they are represented and built by ants. The decision variables implied by this pheromone indicate whether an item is in the solution or not.

The two kinds of pheromone are not mutually exclusive. When a given solution structure represents each solution only once, any pheromone used with or derived from that structure will reflect both solution representation and identity. For instance, $\mathfrak{C} \times \mathfrak{C}$ pheromone for the TSP represents both how solutions are represented (i.e., as permutations)

and the identity of solutions as sets of edges. A number of pheromone representations described in the literature are derived from structural aspects of solutions (see, e.g., Bauer et al., 1999; Colorni et al., 1994; Merkle et al., 2000; Stützle, 1998). For many problems, these pheromones only indirectly identify solutions, an issue which is discussed in Chapter 6.

4.1.2 Atypical Pheromone Representations

An assumption of the notation introduced in this chapter is that a solution is described by a proper subset of the solution characteristics modelled by a pheromone. Accordingly, the pheromone representation may support a number of constructive algorithms for the problem in question in addition to those that build solutions as sequences of solution components. For instance, solutions to the TSP may be built as collections of links while still using a $\mathfrak{C} \times \mathfrak{C}$ pheromone, provided that solution components are taken from $\mathfrak{C} \times \mathfrak{C}$. The notation is thus suited to the majority of pheromone representations used in practice. However, in some ACO algorithms the pheromone representation and constructive process used are either inextricably connected or necessitate special use of pheromone information, and in such situations the notation is less easily applied.

Shortest Common Supersequence Problem

The SCSP (Michel and Middendorf, 1999) requires an unusual constructive approach (in comparison to many of the problems to which ACO has been applied). Although the problem and an ACO algorithm for it were discussed in Section 2.4.6, the description is repeated here given the comparative complexity of the approach used. The SCSP consists of creating a minimum length string of characters from some alphabet Σ such that it is a supersequence of a set L of other strings (i.e., any of the other strings may be produced by deleting characters from the solution). An ACO algorithm for this problem developed by Michel and Middendorf constructs solutions in the following way. Solutions are constructed from the characters in Σ, so $\Sigma \equiv \mathfrak{C}$. Furthermore, unlike the majority of ACO implementations, components in $\mathfrak{C}$ may—generally must— appear multiple times in a solution. Throughout the solution construction process, the algorithm keeps track of how many characters from the start of each string in L have been included in the partial supersequence. The *front* of each string in L is the next character that can be added to the supersequence. At each step, the set of candidate characters consists of the next character to include from each string. A pheromone value is associated with each character in each string, suggesting a pheromone representation of $L \times I$, where I is the set of indices of characters within the strings of L. However, the decision to include a candidate character $c \in \mathfrak{C}$, where $\mathfrak{C}$ is the set of characters, is based on a single pheromone value derived by

summing pheromone values for each $(l, i) \in L \times I$, where i is the position of the next available character from the string l, such that the i^{th} character of l is c.

Given a supersequence produced by the algorithm, pheromone for the characters in each string $l \in L$ is updated by stepping through the construction process used to generate that supersequence to determine which characters were covered at each step, with each string's characters receiving an increase in pheromone in proportion to how early they were included in the supersequence. The pheromone representation is therefore most closely denoted by $\mathfrak{S}^p \times \mathfrak{C}; L \times I$. This differs from more typical pheromone representations in that individual pheromone values have no meaning outside of the constructive process used as each will contribute to the inclusion of a character at some point in a single solution's construction. In more typical pheromone representations the pheromone value associated with each solution characteristic may be considered without its corresponding solution component being added to the solution.

Pheromone for the Direct Assignment of Times in Scheduling Problems

Many scheduling problems to which ACO has been applied construct solutions as sequences of the operations to be scheduled, with a subordinate heuristic performing the final allocation of operations to times in the schedule. This approach is highly intuitive for many such problems as the subordinate heuristic can be extremely simple. For instance, in the JSP and OSP, where the objective is to minimise the makespan of the schedule, scheduling operations as early as the precedence relations established by a solution sequence allow results in the best schedule for those precedence relations. Such problems may also be solved as assignment type problems, where operations are assigned directly to times. An example of such an approach is Randall's (2002b) ACO algorithm for the static single runway aircraft landing problem, introduced in Section 2.4.6. Unlike common assignment problems such as the GAP and QAP, in which $|\mathfrak{C}_{it}| \geq |\mathfrak{C}_{res}|$, representing this problem as an assignment problem results in $|\mathfrak{C}_{it}| \ll |\mathfrak{C}_{res}|$. A further distinguishing characteristic is that resources in this problem are ordinal, while resources in problems like the GAP and QAP are not. Hence, differences between resources are more gradual than in the GAP and QAP, and nearby times are similar in terms of solution cost. As each plane has its own landing window, the set of times $\mathfrak{C}^i_{res}$ available to each plane i is a subset of all available times in the problem $\mathfrak{C}^i_{res} \subset \mathfrak{C}_{res}$. Planes' respective sets of available times form a cover of $\mathfrak{C}_{res}$.

Given that there may be little overlap between planes' respective available times, a typical $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone representation is likely to be highly wasteful of computing resources, with many modelled assignments outside each plane's time window. To deal with this potential problem, Randall's ACO algorithm divides each plane's time window into a

fixed number k of contiguous regions, with pheromone associated with the assignment of a plane to a time in a given region. The observed pheromone is thus $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$, while the underlying pheromone representation is $\mathfrak{C}_{it} \times \mathfrak{C}'_{res}$, where $\mathfrak{C}'_{res}$ is the set of regions. Individual times within the same region are differentiated by the use of heuristic information. While this is not a higher order pheromone, using the notation described in this chapter it is most accurately denoted by $\mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it} \times \mathfrak{C}'_{res}$. Unlike higher order representations, such a pheromone requires a function that maps from each plane's available times to regions in the pheromone representation, $f : \mathfrak{C}_{it} \times \mathfrak{C}_{res} \rightarrow \mathfrak{C}'_{res}$. The mapping is such that if a given plane's time window contains the times $\{t_1, t_2, \ldots, t_n\}$, and the elements of $\mathfrak{C}'_{res}$ are $\{1, 2, \ldots, k\}$, then $f(t \in \{t_1, \ldots, t_{\frac{n}{k}}\}) = 1$, $f(t \in \{t_{\frac{n}{k}+1}, \ldots, t_{\frac{2n}{k}}\}) = 2$ and so on. While this approach is primarily designed to deal with different planes having different time windows, it may also be advantageous given the contiguous nature of resources in the problem, an issue discussed in Section 6.2.4.

Pheromone to Learn Assignment Order in Assignment Problems

The general framework for applying ACO to assignment type problems described by Costa and Hertz (1997) includes provision for two pheromone representations: one to learn what assignments to make and another to learn what order in which to assign items. An example of the former pheromone is the higher order pheromone used by Costa and Hertz in their ACO algorithm for the GCP, which was used to illustrate the general approach.[2] Given that the order in which items are assigned in such problems does not alter the solution represented, a pheromone representation that learns the order in which to assign items does not directly influence which solutions are constructed. However, assignment order clearly has an important impact on the ability of a constructive algorithm to produce good solutions. In effect, such a pheromone can be used to solve a secondary optimisation problem, namely that of the best assignment order to use. Hence the "solution" it helps to produce is an assignment order, the cost of which is determined by the application of ACO to an assignment problem using that order. In general terms, the pheromone could be represented as $(\mathfrak{S}^p \times \mathfrak{C}_{it}; C)$, where C may be any applicable underlying pheromone, which indicates that the assignment order depends on the current partial solution. In practice, any of the pheromones used with permutation problems could be used: $\mathfrak{C}_{it} \times \mathfrak{C}_{it}$, $\mathfrak{C}_{it} \times P$, or $\mathfrak{S}^p \times \mathfrak{C}_{it}; \mathfrak{C}_{it} \times \mathfrak{C}_{it}$ (preceding).

[2] Costa and Hertz (1997) did not use a pheromone to learn a good assignment order, instead using a variety of problem specific heuristics.

4.2 Using Higher Order Pheromone Representations

A number of issues arise when dealing with higher order pheromone representations. Chiefly, these are how the higher order information is used, and the trade-off between the computational overhead associated with the larger pheromone representation and the benefits of using the extra information it provides. The former is discussed in this section, while the latter is discussed in Section 4.2.2 below.

Although higher order pheromone representations have been used in a number of ACO algorithms, there is no single approach to their use. Nevertheless, there are common features of each of the approaches currently described in the literature that allow a general framework to be proposed.

When using a first order pheromone, each constructive step is a competition between individual solution characteristics (and hence between the solution components they implicitly represent). When using a higher order pheromone, the pheromone associated with adding a particular solution component is an aggregate of a number of pheromone values. The notation introduced above partially defines the nature of the relationships between solution characteristics in a higher order pheromone. However, the precise nature of the relationships depends on the problem being solved and the constructive algorithm used. Given a first order solution characteristic c, denote the set of all other single solution components or characteristics to which c is related and which should be used to inform the decision to include c in the current partial solution by C_c. In general, for an n^{th} order pheromone representation it is important to know to which tuples of $(n-1)$ solution components or characteristics a first order characteristic c is related, which is denoted C_c^{n-1}. Given an appropriate definition for C_c^{n-1}, a suitable aggregation function must also be defined, as well as an alternative when $C_c^{n-1} = \varnothing$. Assuming that a solution characteristic c corresponds to a single constructive step (e.g., the addition of a single solution component or a single assignment), and denoting the pheromone associated with adding c to the partial solution $\mathfrak{s}^p$ using an n^{th} order pheromone by $\tau(\mathfrak{s}^p, c, n)$, a generic function for $\tau(\mathfrak{s}^p, c, n)$ where $n \geq 2$ is given by

$$\tau(\mathfrak{s}^p, c, n) = \begin{cases} f(\mathfrak{s}^p, c, \tau_n) & \text{if } \exists\, \tau_n \text{ and } |C_c^{n-1}| > 0 \\ \tau(\mathfrak{s}^p, c, n-1) & \text{otherwise} \end{cases} \tag{4.1}$$

where $\tau_n : C \times C^{n-1} \rightarrow T$ is a function from collections of n solution characteristics to pheromone values, and $f(\mathfrak{s}, c, \tau_n)$ is an aggregation function over the pheromone values associated between c and the elements of C_c^{n-1}, which is discussed in more detail below. Note that the equation is recursive; if C_c^{n-1} is empty or τ_n does not exist then a lower order pheromone representation is sought. To ensure that the recursion defined by Equation 4.1

is well-founded, τ_1 must be defined, either to be a constant value or a separate first order pheromone. In pheromone representations where the elements of C_c^{n-1} are taken from $\mathfrak{s}^p$, early in solution construction $\mathfrak{s}^p$ contains few solution characteristics and it is likely that $C_c^{n-1} = \varnothing$, and hence a lower order pheromone such as τ_1 must be used until the n^{th} order pheromone τ_n can be used. Conceivably, for $n > 1$, if an n^{th} order pheromone is used, $n-1$ other pheromone representations may also be employed to deal with the first $n-1$ steps of solution construction. In practice, most higher order pheromones are only second order, so at most two pheromone representations may be required.

Instances of higher order pheromones are specified by providing definitions for the three components of this general definition, C_c, f and τ_1. The definition of C_c is highly problem specific and closely tied to the way solutions are constructed. Some examples are given below.

A number of options are available for the aggregation function f, four of which are to take the minimum, maximum, mean or sum of the different pheromone values involved, shown in Equations 4.2 through 4.5:

$$f(\mathfrak{s}^p, c, \tau_n) = \arg \min_{c' \in C_c^{n-1}} \tau_n(c, c') \tag{4.2}$$

$$f(\mathfrak{s}^p, c, \tau_n) = \arg \max_{c' \in C_c^{n-1}} \tau_n(c, c') \tag{4.3}$$

$$f(\mathfrak{s}^p, c, \tau_n) = \frac{\sum_{c' \in C_c^{n-1}} \tau_n(c, c')}{|C_c^{n-1}|} \tag{4.4}$$

$$f(\mathfrak{s}^p, c, \tau_n) = \sum_{c' \in C_c^{n-1}} \tau_n(c, c'). \tag{4.5}$$

These four alternative definitions of f are denoted by $min(\tau_n)$, $max(\tau_n)$, $mean(\tau_n)$ and $sum(\tau_n)$ respectively.

The definition of C_c^{n-1} also determines which pheromone values from τ_n are updated by a solution sequence $\mathfrak{s}$. For instance, using a second order pheromone that represents the learned utility of having pairs of solution characteristics $(c_i, c_j) \in C^2$ copresent in a solution, pheromone is updated for all pairs (c_i, c_j) such that $c_i, c_j \in \mathfrak{s}, c_i \neq c_j$. Alternatively, given a solution sequence $\mathfrak{s}$ and a second order pheromone that represents the utility of placing a solution component $\mathfrak{c}$ before certain other solution components $\mathfrak{C}_\mathfrak{c} \subset \mathfrak{C}$, the value of $\mathfrak{C}_\mathfrak{c} \subset \mathfrak{C}$ *when the solution was constructed* must be used to identify which pheromone values to update. Similar examples from the literature are given for both situations below.

4.2.1 Examples

The following examples illustrate the use of higher order pheromones in the literature in relation to Equation 4.1.

Blum (2002b) investigated a first and second order pheromone representation for the KCTP, $\mathfrak{C}$ and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) respectively, where $\mathfrak{C}$ is the set of edges used to construct a k-cardinality tree. Given that $C \equiv \mathfrak{C}$, this second order representation has $C_c = \{c' \mid c' \in \mathfrak{s}^p\}$, with $f = sum(\tau_n)$ and τ_1, used to select the first edge, given by the first order pheromone. Hence, when using the second order pheromone, both the first and second order pheromones are used in solution construction and updated by the solutions produced.

Fenet and Solnon (2003) describe an ACO algorithm for the MCP in which a $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) pheromone is used, where $\mathfrak{C} \equiv C$ is the set of nodes from which a clique must be selected, with $C_c = \{c' \mid c' \in \mathfrak{s}^p\}$ and $f = sum(\tau_n)$. The first node is selected randomly, which is equivalent to $\tau_1 = 1$, as heuristic information is not used.

Costa and Hertz (1997) use a grouping pheromone $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$ in their ACO algorithm for the GCP with $C_{c=(i,r)} = \{j \in \mathfrak{C}_{it} \mid (j,r) \in s_{\mathfrak{s}^p},\ r \in \mathfrak{C}_{res}\}$, $f = mean(\tau_n)$ and $\tau_1 = \tau_0 = 1$.[3] A similar second order pheromone is used by Ducatelle and Levine (2001, 2004) in their $\mathcal{MM}$AS algorithm for the BPP and CStockP, with the exception that $\tau_1 = 1$ while $\tau_0 = \tau_{max} = 20$.

Socha et al. (2002) examined two alternative pheromones for a UCTP, one of which is similar to the grouping pheromone used by Costa and Hertz (1997) and Ducatelle and Levine (2001, 2004). However, rather than learn what events should be scheduled at the same time (equivalent to learning what nodes should be placed in the same colour group in the ACO algorithm for the GCP of Costa and Hertz) this pheromone is used to learn which events should *not* be scheduled at the same time. Consequently, given $C = \{(i,r) \in \mathfrak{C}_{it} \times \mathfrak{C}_{res}\}$ where $\mathfrak{C}_{it}$ is the set of events and $\mathfrak{C}_{res}$ the set of times, $C_{c=(i,r)} = \{j \in \mathfrak{C}_{it} \mid i \neq j,\ (j,r') \notin s_{\mathfrak{s}^p},\ r' \in \mathfrak{C}_{res}\}$ (i.e., the set of unscheduled events), $f = min(\tau_n)$ and $\tau_1 = \tau_{max}$.[4]

ACO algorithms for CSatPs, in which a set of variables must be assigned values from each variable's domain, have used similar higher order pheromones. Roli et al. (2001) describe the use of three alternative pheromones including a second order pheromone $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) where $\mathfrak{C}$ is the set of variable–value assignments. Solnon's (2002) ACO algorithm for the CSatP uses an identical second order pheromone. In both applications,

[3] Note that τ_0 represents the initial pheromone amount while throughout this section τ_n, $n > 0$, is used to denote a function from collections of n solution characteristics to a pheromone value.

[4] τ_{max} is used as Socha et al. (2002) use a $\mathcal{MM}$AS. However, given that the value of τ_1 is only used when there are no unscheduled events remaining, any constant value could be used as the same pheromone value will be used to assess every candidate timeslot.

$C_c = \{c' \mid c' \in \mathfrak{s}\}$ where $C \equiv \mathfrak{C}$ and $f = sum(\tau_n)$. It is not clear in either work what value is given to τ_1 and hence, how pheromone influences the first assignment made.

Schoofs and Naudts (2000) also describe an ACO algorithm for constraint satisfaction for binary CSatPs, in which each constraint involves exactly two variables. Pheromone is associated between pairs of variables and the average pheromone level between a variable i (about to be assigned) and those variables already assigned is used to determine what value to assign to i. However, due to the way pheromone information is updated, it is actually very similar to the approaches of Roli et al. (2001) and Solnon (2002), with the pheromone values used to decide whether to assign a value y_i to the variable i relating to the utility of making that assignment in the preceding generation of ants (B. Naudts, personal communication, 6 December, 2004). Hence, given the set of variables $\mathfrak{C}_{it}$ and the set of values that may be assigned $\mathfrak{C}_{res}$, the pheromone can be described by $C_{c=(i,r)} = \{j \in \mathfrak{C}_{it} \mid (j, r') \in s_{\mathfrak{s}^p},\ r' \in \mathfrak{C}_{res},\ (i, r) \in s', s' \in S^{prev}\}$, where $s_{\mathfrak{s}^p}$ is the partial solution under construction and S^{prev} is the set of solutions from the preceding iteration, $f = mean(\tau_n)$, and $\tau_1 = 1$.

Blum and Sampels's (2002a) pheromone for scheduling problems like the GSP, which learns the relative order of related operations (i.e., those that either require the same machine or are part of the same group), models the first order solution characteristic of adding of a single operation $\mathfrak{c}$ to a partial solution $\mathfrak{s}^p$. However, for simplicity of notation, C_c is denoted by $C_{\mathfrak{c}}$. In this higher order representation, $C_{\mathfrak{c}}$ is the set of unscheduled operations related to operation $\mathfrak{c}$ and $f = min(\tau_n)$. The rationale for using min is that if any one pheromone value is low, then it is likely that $\mathfrak{c}$ should not be added to the solution yet. If $\mathfrak{c}$ has no related operations yet to be incorporated, Blum and Sampels state it is added to the partial solution without considering any other candidate operations, which can be emulated in these equations with $\tau_1 = \infty$.

4.2.2 Notes on Higher Order Pheromone Representation Usage

Defining C_c, f and τ_1 for a Problem

The definition of C_c is problem specific and typically apparent from the higher order solution characteristics being modelled. For instance, if a second order pheromone is used to learn whether pairs of components should be part of the same solution, then intuitively C_c should contain those components already in the partial solution. Alternatively, given a different second order pheromone that models pairs of components that should not be part of the same solution (or where there is a relationship based on the relative order of the pairs of components as in Blum and Sampels (2002a)), and faced with a first order decision about whether to include a candidate characteristic, intuitively C_c should con-

Table 4.2: Sample of ways to customise Equation 4.1.

C_c	f	τ_1	Example(s)
$\in \mathfrak{s}^p$	*sum*	unknown	Roli et al. (2001), Solnon (2002)
		equivalent to 1	Fenet and Solnon (2003)
		1^{st} order pheromone	Blum (2002b)
	mean	1 ($\tau_0 = 1$)	Costa and Hertz (1997), Schoofs and Naudts (2000)
		1 ($\tau_0 = 20$)	Ducatelle and Levine (2001, 2004)
$\notin \mathfrak{s}^p$	*min*	τ_{max}	Socha et al. (2002)
		∞	Blum and Sampels (2002a)

tain only those components that have not yet been added to the partial solution. Both these assertions are based on the rationale that pheromone information is used to determine which solution characteristic(s) to include at each step, rather than which should be ignored (leading to some other solution characteristic(s) being chosen instead).

However, the definitions of f and τ_1 can be somewhat separate from the solution characteristics modelled and so may appear to be arbitrary choices. Nevertheless, based on the examples given in the previous section a number of observations may be made. Table 4.2 categorises the examples based on whether the elements of C_c come from the partial solution or its complement, the aggregation function used and definition of τ_1.

With regards to the aggregation function f, all the cited examples use *min*, *mean* or *sum*, while none uses *max*. The use of the *min* function can be characterised as a cautious approach—any single low pheromone value can in effect veto the first order decision being considered. Conversely, *max* allows any single high pheromone value to make the decision more likely. The functions *sum* and *mean* allow each higher order solution characteristic's pheromone value to influence the first order decision, with the choice of whether to use *sum* or *mean* dependent on the number of higher order solution characteristics available for each candidate first order characteristic (or component). In the examples cited, *sum* is used in all cases where $|C_c| = |C_{c'}| \; \forall \; c \neq c'$ for a fixed partial solution size, while *mean* is used in those cases where this is not the case.

Notably, *min* is used only in those cases where the elements of C_c are not present in the partial solution. In the cited ACO algorithm for the GSP (Blum and Sampels, 2002a), where each solution characteristic indicates the relative order of related operations, the rationale for using *min* is that if any pheromone value is low then there must exist at least one related operation that should be scheduled before the one being considered. Hence, as the relative order of many different subsets of operations is important, taking the mean pheromone value might unfairly represent the true learned utility of placing a particular operation next. The *min* function is also used in the ACO algorithm for the UCTP (Socha et al., 2002), where higher order pheromone values are used to learn which events should

not be placed in the same timeslot. Conceivably, taking the minimum value between the current event and those already assigned a timeslot might produce similar results to considering only those other events yet to be scheduled. However, taking this approach may allow an event to be placed in a timeslot that suits another as yet unscheduled event better and which may increase solution cost if that other event were placed in the same timeslot. Consequently, taking the minimum value may avert such undesirable actions. Thus, in both examples, using *min* in relation to those solution components or characteristics that have yet to be added to a partial solution appears to avoid making those decisions that may force the algorithm to make an inferior decision later in solution construction. In contrast, the use of *sum* and *mean* with pheromone values associated with solution components or characteristics already in a partial solution appears to promote the selection of a solution component or characteristic that is well suited to the existing partial solution.

In those examples where τ_1 is clearly defined and the elements of C_c are taken from the partial solution, the first solution characteristic is chosen either randomly or using a first order pheromone representation when used in conjunction with *sum*, while it is assigned a constant value when used with *mean*. There is no clear relationship between the constant value assigned to τ_1 and τ_0, although differences between the two will conceivably lead to different behaviour in the algorithm. In contrast, where the elements C_c do not come from the partial solution, τ_1 is set to either a high value (τ_{max}) or a candidate component/characteristic is chosen as if it had a high value (such as ∞).

Efficiency Considerations

When implemented, higher order pheromone representations require greater computational resources than their first order counterparts. In terms of storage overhead, n^{th} order pheromones derived from a first order pheromone using Definition 3 grow exponentially with n. However, higher order pheromones are typically only second order, representing a squaring of the number of solution characteristics modelled and hence pheromone values that must be stored. This leads to a relatively small memory overhead.

Nevertheless, higher order pheromone information necessarily takes longer to process than that from a first order representation as multiple pheromone values must be considered for each solution characteristic. This increased computational overhead must be weighed against any potential improvements to the quality of solutions produced by the algorithm. For example, Blum's (2002b) comparative study of first and second order pheromone representations for the KCTP found that, given the same amount of execution time, the latter produces fewer solutions (due to its increased computational overhead). Blum concludes that the first order pheromone is consequently a better choice for this problem. Roli et al. (2001) compare the performance of an ACO algorithm for constraint satisfac-

tion using three alternative pheromone representations, including a first order pheromone that models which assignments should be made and a second order pheromone modelling which pairs of assignments should be made. Both representations performed similarly well (a third representation that models which assignment to make after another assignment performed poorly). However, again due to the increased computational overhead for the second order pheromone, the first order representation is promoted as the best-suited to that problem. Both studies suggests that if a simpler pheromone representation adequately models a problem, it is unnecessary to use a higher order (commonly, a second order) pheromone representation.

4.3 Summary

Despite the increasing range of pheromone representations described in the ACO literature, no consistent language or notation has been developed to describe pheromone representations. This chapter introduced a notation for describing pheromone representations based on the solution characteristics they model. An assumption of the notation is that a solution is described by a proper subset of the solution characteristics modelled by a pheromone. The notation suits the majority of pheromone representations used in ACO implementations, but may also be applied to atypical pheromone representations with little augmentation. Table 4.3 shows the notation applied to a sample of pheromone representations used in the literature, with some additional examples of problems with which certain pheromone representations may be used.

Higher order pheromones require additional specification given the increased number of pheromone values that may be used to influence a single decision. A general framework has been presented into which a number of existing higher order pheromones can be placed. A key feature of any implementation of a higher order pheromone is the function it uses to produce a single pheromone value from the many involved in a decision about a candidate component (or characteristic). The nature of this function appears to be related to whether the solution components or characteristics to which it is related are part of the partial solution or yet to be included. Further investigation is required to determine if general rules can be devised for choosing the most suitable aggregation function for a given higher order pheromone.

In many problems there exists a choice concerning which pheromone representation to use. Given that ACO algorithms use pheromone information to bias constructive decisions, there will necessarily be an interaction between any underlying constructed solution biases and the pheromone representation employed. This interaction is the subject of the next chapter.

Table 4.3: Pheromone representations used in the literature and potentially applicable. Abbreviations used: "rel'd succ." means pairs of related components where their relative order is important; "group" means pairs of items assigned to the same group.

Problem	Pheromone	Example
Subset Problems		
MKP	$\mathfrak{C}$	Leguizamón and Michalewicz (1999)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)	—
SCP	$\mathfrak{C}$	Fiorenzo Catalamo and Malucelli (2001)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	—
KCTP	$\mathfrak{C}$	Blum (2002b)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	Blum (2002b)
MCP	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)	Fenet and Solnon (2003)
SPP	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Maniezzo and Milandri (2002)
Permutation/Routing Problems		
TSP	$\mathfrak{C} \times \mathfrak{C}$	Dorigo et al. (1991)
VRP	$\mathfrak{C} \times \mathfrak{C}$	Bullnheimer et al. (1997)
SOP	$\mathfrak{C} \times \mathfrak{C}$	Gambardella and Dorigo (1997)
Scheduling Problems		
JSP	$\mathfrak{C} \times \mathfrak{C}$	Colorni et al. (1994)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (rel'd succ.)	Blum and Sampels (2002a)
FSP	$\mathfrak{C} \times P$	Stützle (1998)
OSP	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (rel'd succ.)	Blum and Sampels (2002a)
SMTTP[a]	$\mathfrak{C} \times P$	Bauer et al. (1999)
GSP	$\mathfrak{C} \times \mathfrak{C}$	Blum and Sampels (2002a)
	$\mathfrak{C} \times P$	Blum and Sampels (2002a)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (rel'd succ.)	Blum and Sampels (2002a)
Assignment Problems		
QAP	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Maniezzo and Colorni (1999)
FAP	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Maniezzo and Carbonaro (2000)
GAP	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Lourenço and Serra (2002)
Group Assignment Problems		
GCP	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Costa and Hertz (1997)
BPP	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Ducatelle and Levine (2001, 2004)
Timetabling		
UCTP	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Socha et al. (2002)
	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Socha et al. (2002)
Constraint Satisfaction Problems		
	$\mathfrak{C} \times \mathfrak{C}$	Solnon (2000)
	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	Roli et al. (2001)
Others		
AIRLAND	$\mathfrak{C}_{it} \times \mathfrak{C}'_{res}$	Randall (2002b)
SCSP[b]	$\mathfrak{S}^p \times \mathfrak{C}; L \times I$	Michel and Middendorf (1999)
NETSYNTH	$\mathfrak{C}$	Randall and Tonkes (2001)
2DHPPF	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Shmygelska et al. (2002)
BUS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Forsyth and Wren (1997)

[a] SMTTP and variations.

[b] L is the set of strings in the problem, while I is the set of indices of characters in those strings.

Chapter 5

Interaction Between Constructed Solution Biases and Pheromone

Chapter 3 described the biases that may be present in any constructive optimisation algorithm, and which consequently may affect a learning constructive algorithm by exposing it to some solutions more frequently than others. Chapter 4 provided a formal description of the learning mechanism used in ACO, the pheromone representation, and described how different pheromone representations are suited to different COPs. This chapter considers how any underlying constructive biases interact with different pheromone representations. Section 5.1 describes the mechanisms of interaction from two different perspectives. The first considers low level interactions that are a direct result of constructed solution biases. Given that the effects of these underlying biases become difficult to detect as problem size grows, the second discusses higher level interactions that can be linked most directly to the observed behaviour of ACO when using different pheromone representations. The work of Blum (2004) in this area is discussed and augmented by findings made by the study of biases inherent in the constructive process. Section 5.2 is a case study of three commonly used pheromone representations for the JSP, GSP and OSP and reveals the interesting and potentially advantageous structure these problems have when solved by ACO. Section 5.3 briefly describes the interaction between constructed solution bias and pheromone for two other problems whose structure is less obviously exploited as that in the JSP, GSP and OSP. Two problems with no constructed solution biases are also discussed.

5.1 Mechanisms of Interaction

Different pheromone representations interact with the same combinatorial optimisation problem in different ways. This is because the different solution characteristics they model correspond to different patterns of arcs in the construction tree for a given problem. For

instance, a solution characteristic $(i, j) \in \mathfrak{C} \times \mathfrak{C}$ is used in all places that component j appears immediately after component i, while a solution characteristic $(j, k) \in \mathfrak{C} \times P$ is used every time component j is placed at position k. These solution characteristics may correspond to some of the same decisions in the construction tree, but not all. Hence, a single solution characteristic (and its corresponding pheromone value) can appear in many different contexts; the nature of the decisions it affects depends on the number and location of these decisions in the construction tree and so, indirectly, on the decisions made on the paths leading to those points. A certain choice in one context is not necessarily a good choice in all contexts. These differences result in differing reactions to any inherent bias.

The main example COP considered in this chapter is the GSP and it is with regards to this problem that the interaction between bias and pheromone representation is discussed. As the three pheromone representations that have been used in ACO algorithms for this problem—$\mathfrak{C}\times\mathfrak{C}$, $\mathfrak{C}\times P$ and $\mathfrak{S}^p\times\mathfrak{C}; \mathfrak{C}^2$ (related succeeding), where $\mathfrak{C}$ is the set of operations—are referred to frequently throughout this chapter, abbreviated names will be used for each. Hence, $\mathfrak{C} \times \mathfrak{C}$ is denoted by $\mathsf{PH}_{\mathsf{suc}}$, $\mathfrak{C} \times P$ by $\mathsf{PH}_{\mathsf{pos}}$, and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ by $\mathsf{PH}_{\mathsf{rel}}$. These names are similar to those attributed to these pheromone representations by Blum and Sampels (2002a).

5.1.1 Low Level Interactions

Considering an ACO algorithm that uses only pheromone to guide its decisions, a solution characteristic will be reinforced with a frequency related to the distribution of decisions in the construction tree which it influences. If those decisions appear largely in sequences with an inherently high probability, due to a representation or construction bias, then it is more likely to be reinforced than if those decisions were to appear in sequences with an inherently lower probability. The influence (feedback) of a solution characteristic's pheromone value on decisions in subsequent iterations of an ACO algorithm will depend on this same distribution. Moreover, each solution characteristic will not necessarily correspond to the same number of decisions in the construction tree. This last issue becomes critically important as problem size grows and is discussed in the next section.

Consider the `jsp2-2` instance, reproduced with all problem details in Figure 5.1, and its construction tree reproduced in Figure 5.2. The mapping of solution characteristics from $\mathsf{PH}_{\mathsf{suc}}$ to arcs in this construction tree is such that solution characteristic $(1, 2)$ appears in 50% of all sequences, yet due to the inherent construction bias has a 62.5% probability of being updated during the first (essentially undirected) iteration of an ACO algorithm. In contrast, solution characteristic $(3, 2)$ appears in $16\frac{2}{3}\%$ of sequences, but due to the construction bias has only a 12.5% probability of being updated under the same conditions. The fact that these two solution characteristics appear in a different number of sequences

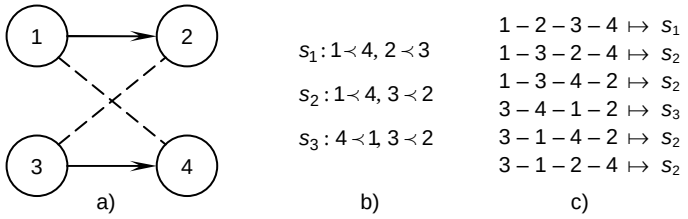


Figure 5.1: A JSP instance described by Blum and Sampels (2002b) a) A small JSP instance, `jsp2-2`, with $\mathcal{O} = \{1, 2, 3, 4\}$, $\mathcal{J} = \{J_1 = \{1, 2\}, J_2 = \{3, 4\}\}$, $1 \prec 2$, $3 \prec 4$, $\mathcal{M} = \{M_1 = \{1, 4\}, M_2 = \{2, 3\}\}$, $p(1) = p(4) = 10$, $p(2) = p(3) = 20$. $i \prec j$ indicates i must be processed before j. b) The three solutions to this problem described in terms of the relative order of operations that require the same machine. $C(s_1) = C(s_3) = 60$, $C(s_2) = 40$. c) The six sequences that may be constructed and the solutions to which they correspond.

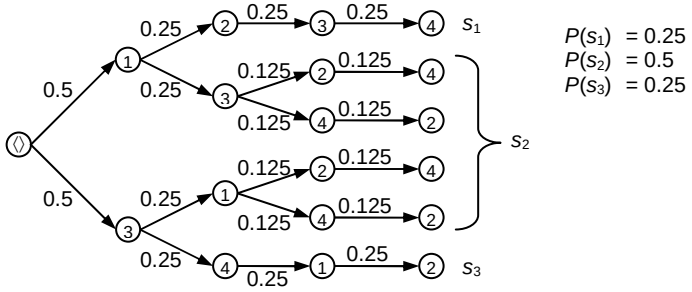


Figure 5.2: Construction tree for `jsp2-2` (see Figure 3.1 for problem description). Arcs are labelled with the probability of their being traversed if using $\text{ACO}_{\text{undir}}$. End points are labelled with the solution represented by that path. Aggregate solution probabilities appear on the right.

is not a result of the representation bias in the JSP—a representation bias does not apply to individual sequences which are by definition unique—but is a feature of the kinds of solution characteristics that $\mathsf{PH}_{\mathsf{suc}}$ models, an issue discussed in more detail in Section 5.2.

As an illustration of the ways in which the solution characteristics from different pheromone representations map onto the construction tree for the `jsp2-2` instance, Figures 5.3 and 5.4 show the patterns of decisions affected using different pheromone representations if pheromone values are updated by sequence $\langle 1, 2, 3, 4\rangle$ (corresponding to solution s_1) and sequence $\langle 1, 3, 4, 2\rangle$ (corresponding to solution s_2) respectively. The three pheromone representations considered are $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$.

Using sequence $\langle 1, 2, 3, 4\rangle$ to update pheromone values from $\mathsf{PH}_{\mathsf{suc}}$ reinforces solution characteristics $(\langle\rangle, 1)$, $(1, 2)$, $(2, 3)$ and $(3, 4)$, thereby making it more likely that operation 1 would be placed first, operation 2 would be placed immediately after operation 1 (regardless of operation 1's position), operation 3 would be placed immediately after operation 2 (regardless of operation 2's position) and operation 4 would be placed immediately after operation 3 (regardless of operation 3's position). If $\mathsf{PH}_{\mathsf{pos}}$ were used, solution character-

istics $(1,1)$, $(2,2)$, $(3,3)$ and $(4,4)$ would all be reinforced, thereby making it more likely that operation 1 would be placed in position 1, operation 2 in position 2, and so on. These pheromone changes only increase the likelihood of producing the same sequence again, as the only decisions associated with these pheromone values that are not part of this sequence are for placing operation 4 in last position when there are no other options. If $\mathsf{PH}_{\mathsf{rel}}$ were used, solution characteristics $(1,4)$ and $(2,3)$ would be reinforced, thereby making it more likely that operation 1 would be scheduled before operation 4 and that operation 2 would be scheduled before operation 3.

Using sequence $\langle 1,3,4,2\rangle$ to update pheromone values from $\mathsf{PH}_{\mathsf{suc}}$ reinforces solution characteristics associated with that sequence as well as one pheromone value associated with the sequence corresponding to solution s_3. If $\mathsf{PH}_{\mathsf{pos}}$ were used, solution characteristics for that sequence are reinforced, as well as one solution characteristic from a different sequence, although the same solution. If $\mathsf{PH}_{\mathsf{rel}}$ were used, solution characteristics $(1,4)$ and $(3,2)$ would be reinforced, thereby making each of the four sequences corresponding to solution s_2 more likely in later iterations. Given that solution s_2 is the optimal solution to this instance, updating $\mathsf{PH}_{\mathsf{suc}}$ with this (optimal) sequence actually makes one of the suboptimal solutions (s_3) more likely, while updating $\mathsf{PH}_{\mathsf{rel}}$ with the same sequence makes only the optimal solution more likely. This interesting feature of the JSP using $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{rel}}$ is discussed in detail in Section 5.2 below.

As discussed in Section 3.5, as problem instance size grows the probability of any one solution becomes extremely small, which makes the effects of any underlying bias difficult to detect. Moreover, distribution of solution characteristics modelled by a pheromone representation becomes increasingly complex. Consequently, low-level interactions between constructed solution biases and pheromone representations may not have immediately identifiable effects on the observed behaviour of an ACO algorithm. The next section considers how the observable behaviour of an ACO algorithm may be predicted using knowledge of the pheromone representation it employs.

5.1.2 High Level Interactions and Competition-Balanced Systems

As problem instance size grows the number of sequences representing solutions $|\mathfrak{S}|$ grows at least as fast as the growth in solutions, which is typically exponential. In problems with a representation bias $|\mathfrak{S}|$ will grow even faster. However, the number of solution characteristics modelled by a pheromone representation $|C|$ grows at a slower rate. For instance, for the MKP, $|\mathfrak{S}| = O(n!)$, while $|C| = O(n)$, where n is the number of items. In the GAP, when using a static assignment order, $|\mathfrak{S}| = O(m^n)$, while $|C| = O(m \cdot n)$, where n is the number of tasks and m is the number of agents. Consequently, the average

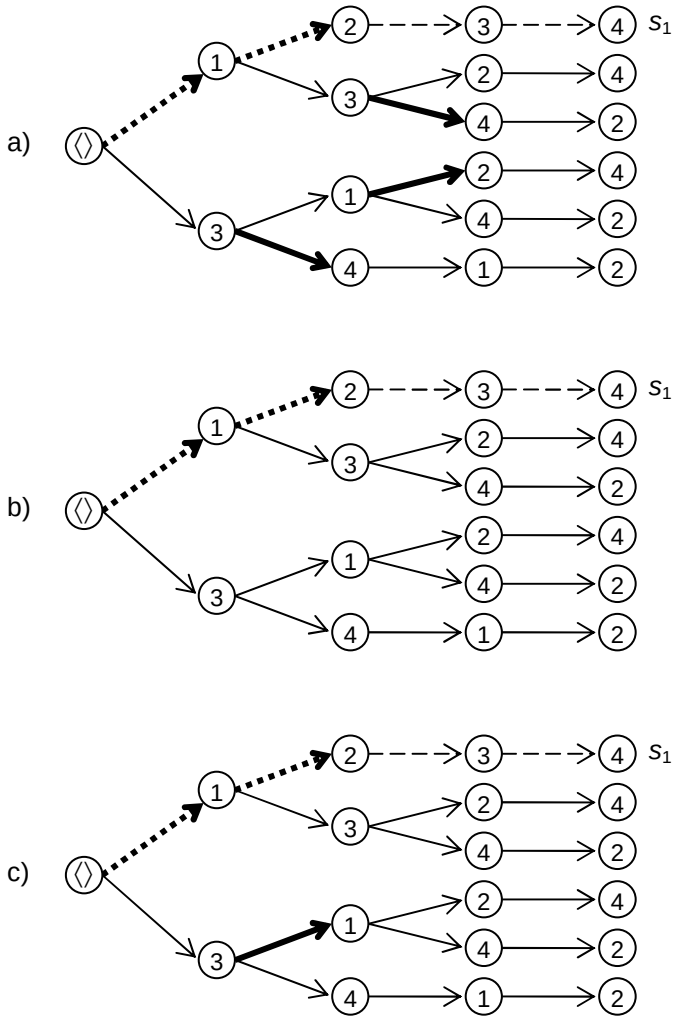


Figure 5.3: Decisions affected by use of sequence $\langle 1, 2, 3, 4 \rangle$ (corresponding to solution s_1 to JSP instance `jsp2-2`) to update pheromone values using a) $\mathsf{PH}_{\mathsf{suc}}$, b) $\mathsf{PH}_{\mathsf{pos}}$, and c) $\mathsf{PH}_{\mathsf{rel}}$. Dashed lines indicate the actual construction path taken. Affected decisions (i.e., arcs) are shown in bold; where only one choice is available the corresponding pheromone value has no influence and hence that arc does not appear in bold.

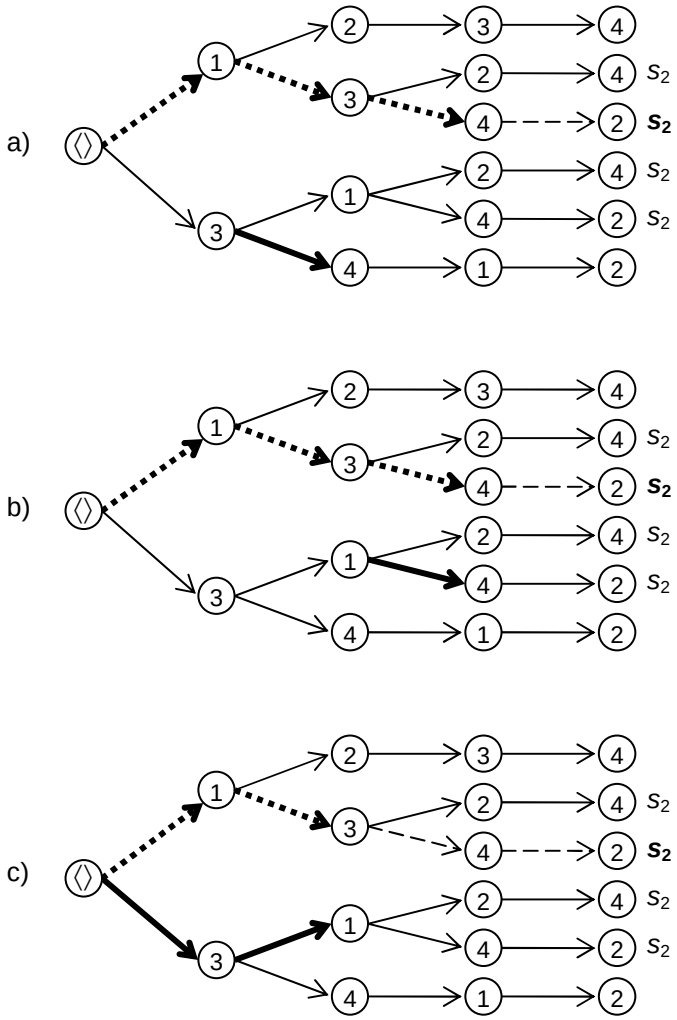


Figure 5.4: Decisions affected by use of sequence $\langle 1, 3, 4, 2 \rangle$ (corresponding to solution s_1 to JSP instance `jsp2-2`) to update pheromone values using a) $\mathsf{PH}_{\mathsf{suc}}$, b) $\mathsf{PH}_{\mathsf{pos}}$, and c) $\mathsf{PH}_{\mathsf{rel}}$. Dashed lines indicate the actual construction path taken. Affected decisions (i.e., arcs) are shown in bold; where only one choice is available the corresponding pheromone value has no influence and hence that arc does not appear in bold.

number of sequences in which each solution characteristic appears (denoted by $|\mathfrak{S}_c|$) grows with instance size and any differences between $|\mathfrak{S}_c|$ for different solution characteristics will influence which are most likely to be reinforced by solutions produced by the algorithm.

In a separate effort to understand the way different pheromone representations will behave for a given problem, Blum (2004) introduces the concept of a *competition balanced system*. In terms of ACO this is defined as a pheromone representation consisting of solution characteristics that appear in the same number of sequences produced by the algorithm. Blum (2004, p.74) provides the following definition of a competition-balanced system (CBS) (explanatory notes appear in square brackets):

> Given a model $\mathcal{P}$ of a CO [combinatorial optimisation] problem, we call an ACO algorithm *together with* the instance P of $\mathcal{P}$ to which it is applied a *competition-balanced system (CBS)*, if the following holds: Given a feasible partial solution $\mathfrak{s}^p$ and the set of solution components [equivalent to a solution characteristic as described in Chapter 4] $\mathfrak{N}(\mathfrak{s}^p)$ that can be added to extend the partial solution $\mathfrak{s}^p$, each solution component [i.e., characteristic] $\mathfrak{c} \in \mathfrak{N}(\mathfrak{s}^p)$ is a component of the same number of feasible solutions (in terms of sequences built by the algorithm) as each other solution component $\mathfrak{c}' \in \mathfrak{N}(\mathfrak{s}^p), \mathfrak{c} \neq \mathfrak{c}'$.

Note that the term *solution component* in this definition is equivalent to the definition of a *solution characteristic* defined in Chapter 4. If a pheromone model applied to a particular problem instance is not a CBS, bias may be observed. The issue of what effects this bias may have is discussed in Sections 5.2 and 5.3 below.

The definition of a CBS does not explicitly incorporate the influence of the underlying constructed solution biases responsible for the low-level interactions described in the previous section. However, the presence of such biases is the underlying determinant of whether or not a pheromone representation is a CBS. Furthermore, it is possible for a pheromone representation to be a CBS, yet not be free from bias.

Theorem 1 *A CBS (as defined by Blum (2004)) is not necessarily free from solution bias.*□

Proof When constructing solutions to the OSP as permutations of operations, both $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{pos}}$ are CBSs, as each solution characteristic appears in the same number of sequences (since all permutations of operations are feasible). However, the OSP also has a representation bias and hence some solutions will be represented by larger sets of solution characteristics, corresponding to each sequence representing those solutions. The resulting pheromone–problem combination is therefore not free from solution bias. ■

The following Lemmas demonstrate that, for all practical purposes in other circumstances, constructed solution biases will prevent any pheromone representation from being a CBS.

Lemma 1 *A representation-oriented pheromone representation applied to a COP with a construction bias cannot be a CBS.* □

PROOF A representation-oriented pheromone representation (as defined in Section 4.1.1) contains solution characteristics that describe each sequence uniquely. Consequently, in order that such a pheromone representation be a CBS, each sequence must represent a feasible solution. This criterion is not met by a problem with a construction bias. ■

Lemma 2 *An identity-oriented pheromone representation applied to a COP with a representation bias cannot be a CBS.* □

PROOF An identity-oriented pheromone representation (as defined in Section 4.1.1) contains one set of solution characteristics for each distinct solution. In order that such a pheromone representation be a CBS, each solution must be represented by the same number of sequences. This criterion is not met by a problem with a representation bias. ■

Notably, while the representation-oriented $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{pos}}$ applied to the OSP are CBSs (although not free from the underlying representation bias), the identity-oriented $\mathsf{PH}_{\mathsf{rel}}$ applied to the same problem is not a CBS.

Lemma 3 *An identity-oriented pheromone representation applied to a COP with a construction bias cannot be a CBS.* □

PROOF Given Lemmas 1 and 2, the Lemma need only be proved for those cases where the pheromone is not also representation-oriented (see Lemma 1) and where each solution is represented by the same number of sequences (see Lemma 2). The latter implies that each solution is described by the same number of solution characteristics (otherwise the problem must have a representation bias due to the different number of permutations of solution characteristics that would result if solutions were described by a variable number of characteristics). In order that such a pheromone be a CBS, every solution characteristic must appear in the same number of combinations (of solution characteristics) as every other. A construction bias implies that this criterion cannot be met, as some solution characteristics must appear less often than others for the degree of nodes within each level of the construction tree to be non-uniform. ■

Based on the above, it is possible to state the following Theorem.

Theorem 2 *If a constructive algorithm applied to a COP has a construction bias then there does not exist a representation- or identity-oriented pheromone representation that is a CBS.* □

PROOF The statement is a direct consequent of Lemmas 1 and 3. ■

Given these findings, it is plausible that if a problem has no constructed solution bias then for all practical cases any pheromone representation for it will be a CBS.

The concept of a CBS helps to identify those situations where a bias may be observed in the frequency with which different pheromone values are updated by solutions produced by an ACO algorithm, without requiring full knowledge of the distribution of solution characteristics in the construction tree. Furthermore, the existence of an underlying construction bias—a feature which is typically easily identified from knowledge of the problem and the constructive algorithm used—can indicate that no pheromone representation can be a CBS for a particular problem. However, neither of these can necessarily be used to predict exactly *how* a particular pheromone will be biased.

Only with specialised insight into the nature of a problem and of the likely frequency of solution characteristics can estimates be made of the number and *kind* of sequences each characteristic will appear in. Estimates of the likely bias towards reinforcing particular solution characteristics, including taking account of underlying constructed solution biases, may only proceed if the problem in question has clearly identifiable types of solution, with which those solution characteristics are strongly associated, and a bias favouring one type over another. The following case study describes a problem which has these qualities and hence about which predictions may be made by studying its low-level constructed solution biases as well as the high-level feature of the number of sequences in which each solution characteristic appears.

5.2 Case Study: Pheromone Representations for the JSP, GSP and OSP

The following case study illustrates the high level interactions that can occur between pheromone and a problem's construction tree. In empirical work conducted by Blum and Sampels (2002b) and in the current investigation, $\mathsf{PH}_{\mathsf{suc}}$ pheromone has been found to perform poorly on permutation scheduling problems such as the JSP and GSP. Its performance is worst on the JSP, but improves as problem constraints are eased so that its performance, while not the best, is quite good on the OSP. Investigation of the mechanisms underlying this behaviour reveals that these problems have an interesting structure that, at the construction tree level, introduces a bias towards good solutions and, when using different pheromone representations, can serve to bias an ACO algorithm either towards or away from good solutions. The structural aspects of construction trees for these problems are considered first.

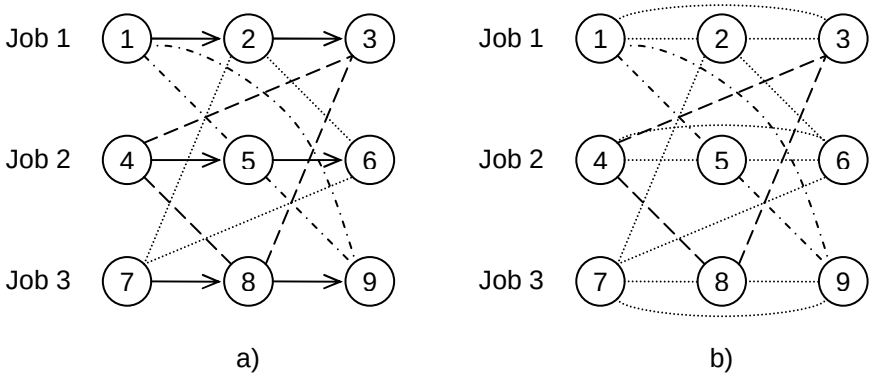


Figure 5.5: Nine operation, three job, three machine JSP and OSP instances: `jsp3-3` and `osp3-3`. In both instances $\mathcal{O} = \{1, \ldots, 9\}$, $\mathcal{J} = \{J_1 = \{1, 2, 3\}, J_2 = \{4, 5, 6\}, J_3 = \{7, 8, 9\}\}$, $\mathcal{M} = \{M_1 = \{1, 5, 9\}, M_2 = \{2, 6, 7\}, M_3 = \{3, 4, 8\}\}$, $p(1) = p(5) = p(9) = 10$, $p(2) = p(6) = p(7) = 20$, $p(3) = p(4) = p(8) = 30$.

Blum and Sampels (2002b) and Blum (2004) have found that sequences corresponding to poor solutions tend to have runs of operations from the same job. They measure this characteristic of sequences by introducing a *line scheduling factor*,[1] given by

$$f_{ls}(\mathfrak{s}) = \frac{\sum_{i=1}^{|\mathcal{O}|-1} \delta(\mathfrak{s}, i)}{|\mathcal{O}| - |\mathcal{J}|} \tag{5.1}$$

where $\mathfrak{s}[i]$ is the operation in the i^{th} position of $\mathfrak{s}$, and $\delta(\mathfrak{s}, i) = 1$ if $\mathfrak{s}[i]$ belongs to the same job as $\mathfrak{s}[i+1]$, 0 otherwise. Hence, the value of f_{ls} is in $[0, 1]$, where 1 indicates that all operations for each job are contiguous, while 0 indicates that no pairs of operations from the same job are adjacent in the sequence.

Sequences with a high line scheduling factor generally correspond to poor solutions to these problems. Intuitively this is to be expected as good schedules allow operations from different jobs to run in parallel. A sequence in which all operations from one job appear in a contiguous group can produce a schedule which contains lengthy delays for other jobs' operations, which must wait for operations from the first job to finish. This intuitive claim is confirmed by empirical results. Figure 5.5 describes a JSP and an OSP instance, both with nine operations, three jobs and three machines. The top row of Figure 5.6 plots the mean f_{ls} value of sequences for each solution against the cost of the solution represented for these two instances.

In GSP instances that are not OSP instances, a construction bias always exists in favour of solutions with a high line scheduling factor. Take for example the JSP. In a JSP with n jobs, n operations are available to be added to the sequence at each step (i.e., one from each job) until all the operations from one of the jobs have been added to the sequence, after

[1] Blum (2004) refers to this measure simply as a *sequencing* factor, denoted by f_{seq}.

which $n-1$ operations are available. As each job's set of unscheduled operations becomes empty, the number of available operations becomes smaller. Thus, selecting an operation from the same job as that last added to the sequence decreases the number of steps until that job's set of unscheduled operations becomes empty and consequently makes it more likely that the same will have to be done with operations from other jobs later in solution construction. Consider a JSP with n jobs of m operations each. A sequence with $f_{ls} = 1$ can be produced on a path with m steps of n options, followed by m steps of $n-1$ options, m steps of $n-2$ options and so on, finishing with m steps of 1 option only. Denote this sequence by $\mathfrak{s}^{f_{ls}=1}$. Consider an alternative sequence constructed by selecting an operation from each job in a round-robin fashion, which accordingly has $f_{ls} = 0$. The path for such a sequence will have $(m-1) \cdot n + 1$ steps at which every job has at least one remaining operation to be scheduled, followed by $n-1$ steps with decreasing numbers of options, $n-1$, $n-2$, ..., 1, as each job's set of unscheduled operations becomes empty. Denote this sequence by $\mathfrak{s}^{f_{ls}=0}$.

Given that the probability of a sequence being produced by $\text{ACO}_{\text{undir}}$ is the inverse of the product of the number of options at each step, the respective probabilities of $\mathfrak{s}^{f_{ls}=1}$ and $\mathfrak{s}^{f_{ls}=0}$ are

$$P(\mathfrak{s}^{f_{ls}=1}) = \frac{1}{\prod_{i=0}^{n-1}(n-i)^m} \tag{5.2}$$

and

$$P(\mathfrak{s}^{f_{ls}=0}) = \frac{1}{n^{(m-1)\cdot n+1} \cdot (n-1)!} \ . \tag{5.3}$$

In general, $P(\mathfrak{s}^{f_{ls}=1}) > P(\mathfrak{s}^{f_{ls}=0}) \quad \forall\, m, n > 1$.

Except for the OSP, where all sequences have equal probability, a sequence with a high f_{ls} value typically has a higher probability. The disparity in probability between sequences with $f_{ls} = 1$ and those with $f_{ls} = 0$ is greatest on the JSP and diminishes as operation precedence constraints are eased (i.e., in GSP instances with groups containing increasing numbers of operations), becoming zero in OSP instances. Thus sequences corresponding to poor solutions, which typically have a high line scheduling factor, are likely to have a relatively high probability of being found in the construction trees for JSP and GSP instances (excluding OSP instances). This is illustrated in the second row of Figure 5.6.

However, solutions represented by sequences with predominantly high line scheduling factors are generally represented by fewer sequences, across all GSP instances. The third row of Figure 5.6 plots the mean line scheduling factor of solutions' sequences against the number of sequences representing that solution. Intuitively, sequences with a high f_{ls} value can tolerate only small perturbations before the solution represented changes. Certainly, in the JSP, a sequence with $f_{ls} = 1$ can only be altered slightly before the relative order of related operations is changed and the sequence represents a different

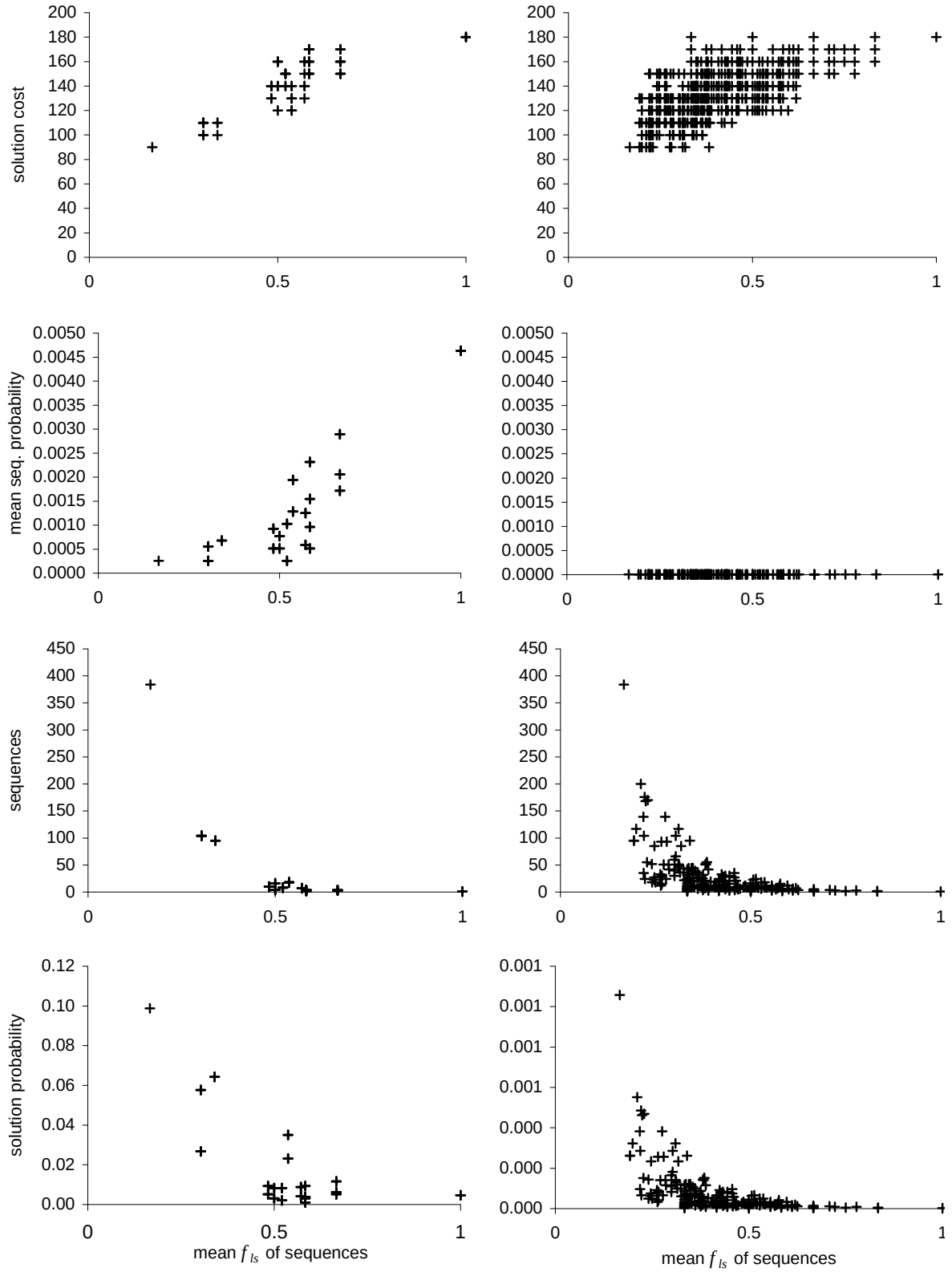


Figure 5.6: Mean f_{ls} values of solutions' sequences against: solution cost (top row); mean probability of solutions' sequences (second row); number of sequences per solution (third row); and solution probability (bottom row) for `jsp3-3` (left) and `osp3-3` (right).

solution. Accordingly, the lower the line scheduling factor, the easier it is to perturb the sequence without changing the relative order of related operations. This suggests that low cost solutions, which are generally represented by sequences with a low f_{ls} value, are overrepresented in the construction tree.

Indeed, the representation bias, which typically favours good solutions to these problems, can overwhelm the construction bias that typically favours poorer solutions. The fourth row of Figure 5.6 plots the mean line scheduling factor of solutions' sequences against the overall probability of finding that solution using $\text{ACO}_{\text{undir}}$.

While the effects of these low level biases are not easily observed on large instances, the mechanisms that produce them also determine how the different pheromone representations available for this problem behave. The remainder of this section focuses on the way $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{rel}}$ pheromones interact with these problems. $\mathsf{PH}_{\mathsf{pos}}$ is also discussed, although it shows a less obvious bias than the other two. Blum and Sampels (2002b) observed high f_{ls} values (up to 1) for sequences produced by $\mathsf{PH}_{\mathsf{suc}}$ applied to GSP instances other than the OSP. In contrast, f_{ls} values when using $\mathsf{PH}_{\mathsf{rel}}$ were consistently low (less than 0.1) across the JSP, GSP and OSP. This result has been found across a range of instances of varying size. A high f_{ls} value was found to be a good predictor of high cost regardless of problem size. Figure 5.7 plots f_{ls} values against solution cost for sequences produced by ACO algorithms using $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{rel}}$ and $\mathsf{PH}_{\mathsf{pos}}$ applied to the `la38` JSP instance.[2] Data were collected by sampling every 100^{th} sequence produced by an ACO algorithm producing a total of 30,000 sequences.[3] As Figure 5.7 shows, while $\mathsf{PH}_{\mathsf{pos}}$ produces many poor solutions to this instance it also produces solutions better than those found by $\text{ACO}_{\text{undir}}$.

Notably, the probability of producing these solutions if using $\text{ACO}_{\text{undir}}$ follows the same pattern established for those smaller instances for which complete exploration of the construction tree is possible. Figure 5.8 plots f_{ls} values against the expected probability of the corresponding sequences being produced by $\text{ACO}_{\text{undir}}$ for the same sample of solutions used in Figure 5.7.

An insight into the strong bias $\mathsf{PH}_{\mathsf{suc}}$ exhibits towards solutions with a high f_{ls} value can be obtained in a number of ways. The first is in terms of CBSs. In the OSP, every solution characteristic from $\mathsf{PH}_{\mathsf{suc}}$ appears in the same number of sequences (i.e., solution representations), so this pheromone is a CBS. In more constrained GSP instances, and especially in the JSP, the use of $\mathsf{PH}_{\mathsf{suc}}$ pheromone ceases to produce a CBS. Indeed, when using $\mathsf{PH}_{\mathsf{suc}}$ pheromone in such instances, solution characteristics corresponding to placing two operations from the same job in succession appear in proportionally more sequences than those for which it is not the case. In contrast, solution characteristics from $\mathsf{PH}_{\mathsf{rel}}$

[2]This instance is part of a benchmark JSP set described by Lawrence (1984).

[3]The actual algorithm used is a modification of Ant Colony System from which heuristic information and its greedy bias (q_0) have been removed.

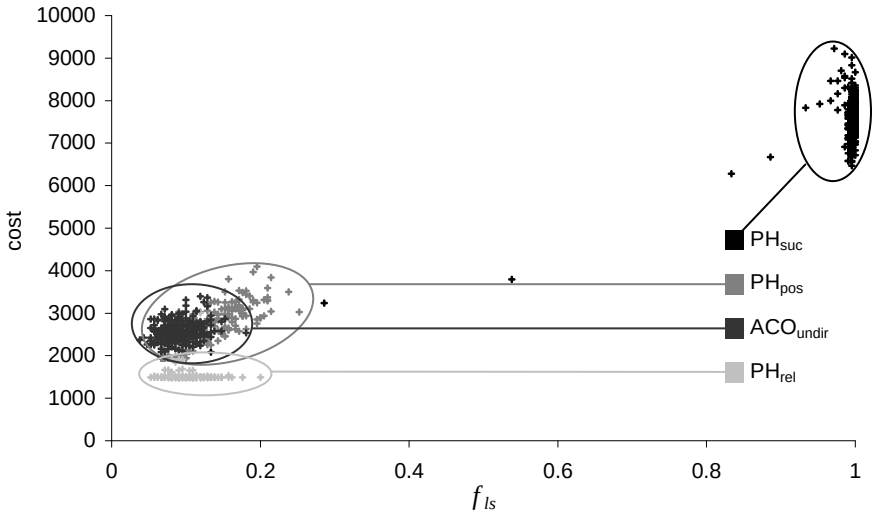


Figure 5.7: f_{ls} values of samples of 300 sequences produced by $\text{ACO}_{\text{undir}}$ and ACO with $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ against cost of solutions represented for `la38` JSP. The regions within which each pheromone's data fall are enclosed by labelled ellipses. The outliers between the data points for $\mathsf{PH}_{\mathsf{suc}}$ and that of the other approaches belong to $\mathsf{PH}_{\mathsf{suc}}$.

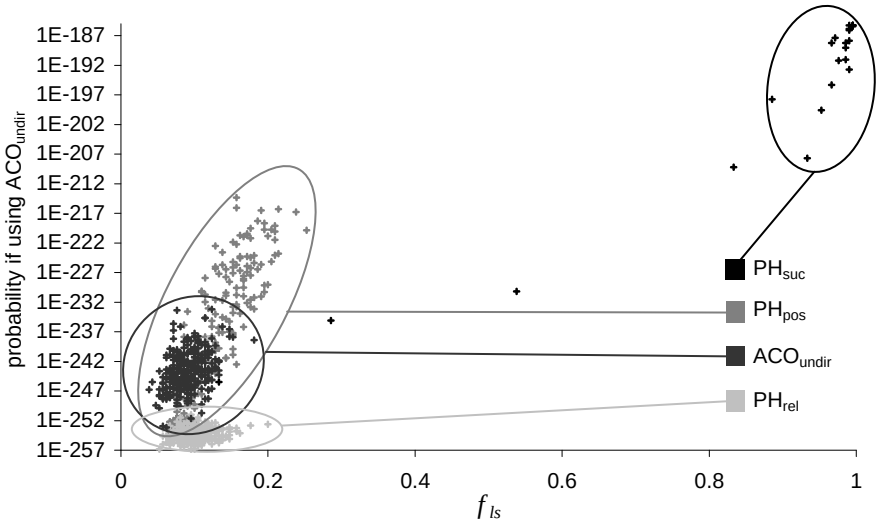
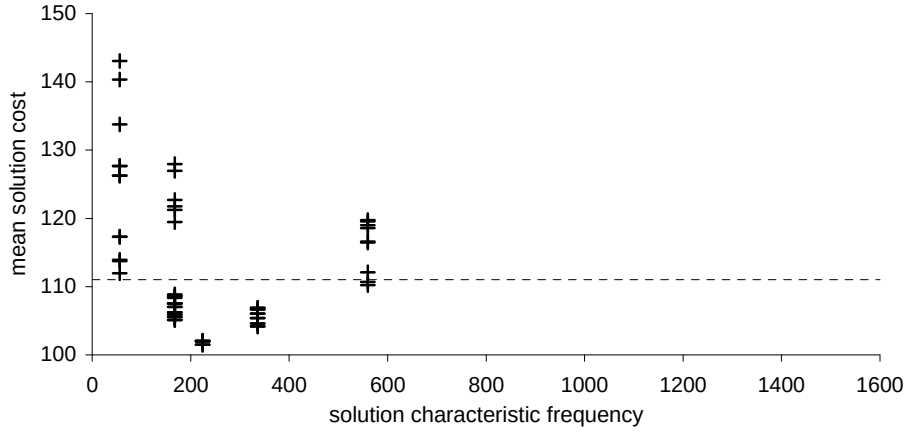


Figure 5.8: f_{ls} values of samples of 300 sequences produced by $\text{ACO}_{\text{undir}}$ and ACO with $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ against probability of sequences if produced by $\text{ACO}_{\text{undir}}$ for `la38` JSP. The regions within which each pheromone's sequences fall are enclosed by labelled ellipses. The outliers between the data points for $\mathsf{PH}_{\mathsf{suc}}$ and that of the other approaches belong to $\mathsf{PH}_{\mathsf{suc}}$.

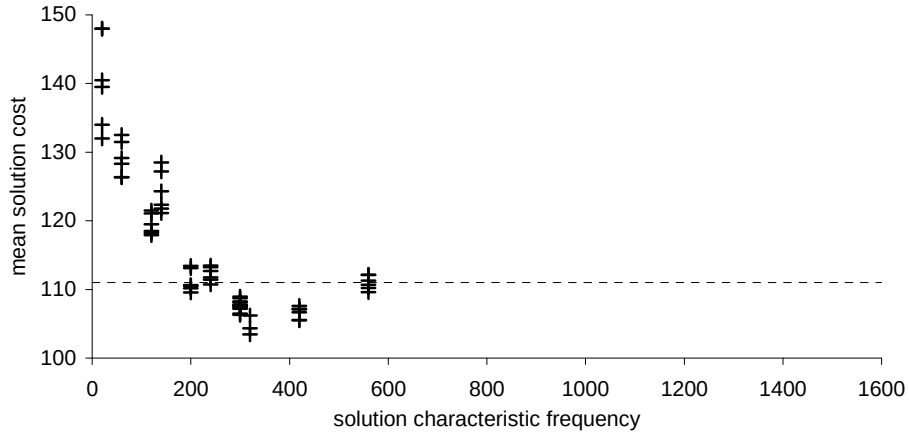
that are associated more strongly with sequences with a low f_{ls} value appear in a greater number of sequences than those characteristics that are not. Thus, in problems where a high f_{ls} value is strongly predictive of a high solution cost, use of $\mathsf{PH}_{\mathsf{suc}}$ will make good solutions increase the pheromone associated with poor solutions, whereas use of $\mathsf{PH}_{\mathsf{rel}}$ will result in even poor solutions increasing pheromone associated most strongly with good solutions. There is no evidence that solution characteristics from $\mathsf{PH}_{\mathsf{pos}}$ that may lead to poor solutions appear more frequently than those associated with good solutions. Indeed, the results of studies described in Chapter 7 show that its performance on this JSP instance is not indicative of its performance on other JSP instances.

Figure 5.9 shows plots of the number of sequences in which each solution characteristic appears against the mean cost of the solutions represented by those sequences, for each of $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ applied to the `jsp3-3` instance. The mean cost of all sequences is shown as a dashed line. All three plots show some commonality in structure, with solution characteristics that appear in predominantly poor solutions having the lowest frequency and solution characteristics that appear in predominantly good solutions having a frequency intermediate between the extremities. This relatively small degree of similarity is to be expected given that the different solution characteristics modelled will have some overlap in the features of solutions they describe. However, at the extremities of frequency, the distribution for $\mathsf{PH}_{\mathsf{suc}}$ shows a range of solution costs associated with solution characteristics, while $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ have much tighter ranges. That is, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ show a stronger relationship between frequency of usage and associated solution cost. Moreover, those solution characteristics that correspond to the best solutions have a frequency of less than half the maximum in $\mathsf{PH}_{\mathsf{suc}}$, just over half the maximum in $\mathsf{PH}_{\mathsf{pos}}$ and 84% of the maximum in $\mathsf{PH}_{\mathsf{rel}}$. Additionally, the most frequently occurring solution characteristics in $\mathsf{PH}_{\mathsf{rel}}$ are also associated with good solutions. Similar results were obtained for other small instances.

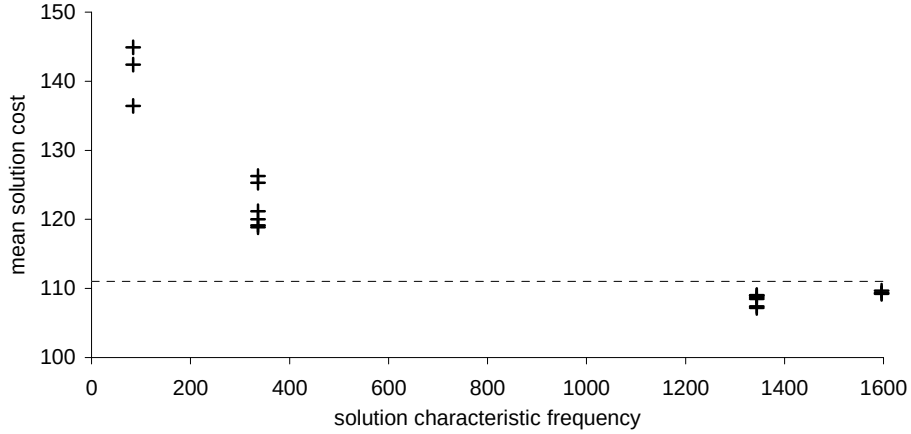
Consideration of the topologies of construction trees for these problems reveals why the solution characteristics from $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{rel}}$ are so strongly biased towards different kinds of sequences and hence, solutions. Given that selecting an operation from the same job as that most recently selected decreases the likelihood that successive pairs of operations placed later will be selected from different jobs, those solution characteristics from $\mathsf{PH}_{\mathsf{suc}}$ that correspond to placing successive operations from different jobs are also less likely to appear in those sequences. In contrast, partially constructed sequences with a low f_{ls} value restrict the set of available operations less and so still allow successive operations from the same job to be placed. Thus, the same mechanism that introduces a construction bias (which has little detectable effect on larger instances) does have an effect on the distribution of solution characteristics from $\mathsf{PH}_{\mathsf{suc}}$ in the construction tree. Conversely,



(a) $\mathsf{PH}_{\mathsf{suc}}$



(b) $\mathsf{PH}_{\mathsf{pos}}$



(c) $\mathsf{PH}_{\mathsf{rel}}$

Figure 5.9: Frequency of usage of solution characteristics against the mean cost of solutions they describe for `jsp3-3` instance, for each of (a) $\mathsf{PH}_{\mathsf{suc}}$, (b) $\mathsf{PH}_{\mathsf{pos}}$ and (c) $\mathsf{PH}_{\mathsf{rel}}$. The mean cost of all sequences is shown as a dashed line.

many of the operation precedence relationships established by sequences with a high f_{ls} value are largely restricted to those sequences and are not present in those sequences that may be perturbed while maintaining the solution represented. Sequences with a high f_{ls} value will still contain some of those operation precedence relationships that appear in better solutions and so overall the number of sequences that these precedence relationships appear in is relatively high. The representation bias in these problems serves to accentuate the effect, as all sequences for a single solution exhibit the same solution characteristics in $\mathsf{PH}_{\mathsf{rel}}$.

The behaviour of $\mathsf{PH}_{\mathsf{pos}}$ is less easy to explain, either in terms of CBSs or by consideration of construction tree topology. Analysis of the frequency of solution characteristics associated with poor solutions reveals they are less likely to be reinforced than those associated with better solutions. Furthermore, unlike $\mathsf{PH}_{\mathsf{suc}}$, the placement of an operation from the same job as the last operation does not relate to a single solution characteristic, as each operation will have a range of possible positions in the sequence. However, $\mathsf{PH}_{\mathsf{pos}}$ clearly produces many solutions of poorer quality than those produced by $\mathrm{ACO}_{\mathrm{undir}}$. Possibly this is because $\mathsf{PH}_{\mathsf{pos}}$ *can* learn to place an operation in a location likely to be immediately after an operation from the same job, so a chance encounter with a poor solution can make similar poor solutions more likely. In contrast, $\mathsf{PH}_{\mathsf{rel}}$ has no solution characteristics that model such a choice.

5.3 Pheromone and Bias Interactions in Other Problems

In the JSP, examination of the frequency of individual solution characteristics against the mean value of solutions they represent reveals that certain solution characteristics are strongly associated with solutions of a particular cost. Indeed, using $\mathsf{PH}_{\mathsf{rel}}$ with the `jsp3-3` instance, the relationship between the number of sequences in which a solution characteristic appears and the mean cost of solutions it describes approaches a linear one, with a correlation coefficient of -0.87. However, the frequency of solution characteristics compared with the mean value of solutions they represent is insufficient to explain the strong bias $\mathsf{PH}_{\mathsf{suc}}$ exhibits towards poor solutions. It is only explained by the fact that making one "poor" decision makes other decisions of a similar kind more likely.

The obvious structure in the JSP (and other non-OSP GSP instances) raises the question of whether other problems also have an exploitable structure. Consequently, a similar examination of pheromone representations for the MKP and GAP was carried out. The TSP and QAP, which are without constructed solution biases, are also discussed.

5.3.1 MKP

In the MKP, it is expected that there exists a strong negative correlation between an item's resource requirements (often referred to as *weight*) and its frequency of inclusion in solutions. Figure 5.10a plots the mean proportion of knapsack capacity used against the frequency of inclusion in sequences for each item in the `mknap1-15item` instance. A linear regression of the data yields a correlation coefficient of -0.97. Linear regression analyses of the same data for the `mknap1-6item` and `mknap1-10item` instances also show strong negative correlations of $\rho = -0.91$ and -0.94 respectively. Accordingly, it would be expected that solution characteristics that correspond to the inclusion of items with greater resource requirements would occur less frequently in solutions than those with lower requirements. Considering solution characteristics from the $\mathfrak{C}$, $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), $\mathfrak{C} \times \mathfrak{C}$, and $\mathfrak{C} \times P$ pheromone representations applied to the `mknap1-15item` instance, linear regression analyses of solution characteristic frequency against the resource requirements of the item(s) included by each characteristic show strong negative correlations of $\rho = -0.95$, $\rho = -0.96$, $\rho = -0.96$, and $\rho = -0.93$ respectively.[4]

In the MKP, item weight and value may be related. The correlations between the two for the `mknap1-6item`, `mknap1-10item` and `mknap1-15item` instances are $\rho = 0.742$, $\rho = 0.909$ and $\rho = 0.828$ respectively. Consequently, it would be expected that solution characteristics associated with high value solutions would also appear in fewer sequences and so be reinforced less frequently. Figure 5.10 parts (b) through (e) plot the frequency of solution characteristic usage against the mean value of solutions represented for the `mknap1-15item` instance for the $\mathfrak{C}$, $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ pheromone representations. Examination of the plots shows that the data fall into three distinct groups. One group, with a low frequency but high mean solution value, consists solely of data for solution characteristics associated with including the heaviest, but most valuable item in the problem. Another, with a low frequency and low mean solution value, consists solely of data for solution characteristics associated with including the next heaviest item, which has a value slightly above average. The other group consists entirely of data for solution characteristics associated with including other items in the problem and is located around the mean value of all sequences. This suggests that although solution characteristics associated with items that are both valuable and resource intensive may be updated less frequently, overall the majority of solution characteristics appear in so many solutions (with a correspondingly large range of values) that any bias towards one characteristic may not necessarily bias the search towards good or bad solutions.

The relative performance of ACO using the four pheromone representations considered

[4]While $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ pheromone representations are not typically used with subset problems, they are supported by any ACO algorithm that constructs solutions as a sequence of the items included.

here is compared in Chapter 7.

5.3.2 GAP

A similar analysis to that above was made of the GAP. As with items in the MKP, assignments with relatively high capacity utilisation would be expected to occur less frequently in the solutions that may be produced. Linear regression of the proportion of an agent's capacity used by an assignment against that assignment's frequency of use, for `gap1-1` and `gap2-1` instances, shows moderate correlations of $\rho = -0.73$ and $\rho = -0.79$ respectively. Figure 5.11a plots the proportion of agent capacity used against the frequency of inclusion in sequences for each assignment in the `gap2-1` instance. Given the GAP has no representation bias and assuming that infeasible solutions are not allowed to update pheromone values, the frequency of usage of solution characteristics from a $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone representation is exactly as Figure 5.11a shows.

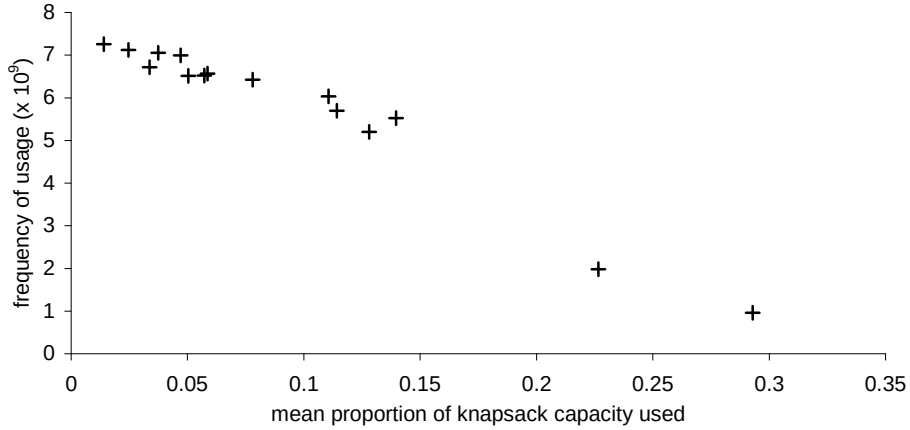
Unlike the MKP, high resource requirements are not associated with high value.[5] Linear regression of agent capacity consumed by an assignment and the value of that assignment produces no correlation in either the `gap1-1` or `gap2-1`, with $\rho = -0.01$ and $\rho = -0.02$ respectively. Linear regression of the same relationship for the `gap12-1` instance studied in Chapter 7 also shows no correlation, with $\rho = -0.08$. Thus, although solution characteristics corresponding to assignments that consume a relatively small proportion of an agent's capacity are made more frequently than those that consume more, the absence of a clear relationship between capacity consumed and value suggests that solution characteristics are unlikely to be strongly associated with solutions of a particular value. Figure 5.11 parts (b) and (c) plot the frequency of solution characteristic usage against the mean value of solutions represented for the `gap2-1` instance for $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ pheromone representations, using the static, fixed assignment order described in Chapter 3. Although some solution characteristics with a low frequency of use are clearly associated more strongly with either high or low value solutions, as a solution characteristic is used in more solutions, the mean value of those solutions approaches the mean value of all solutions.

The relative performance of ACO using $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ pheromone representations is compared in Chapter 7.

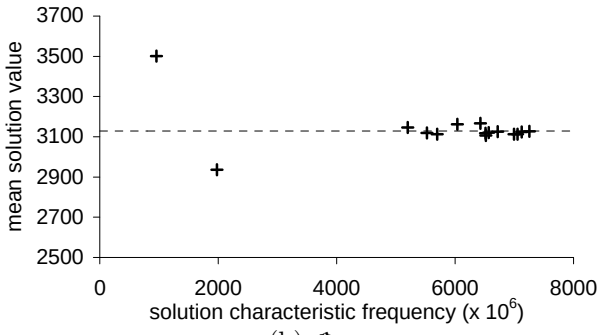
5.3.3 TSP and QAP

The TSP and QAP are both without representation or construction bias and consequently it may be expected that all available pheromone representations should be CBSs for these

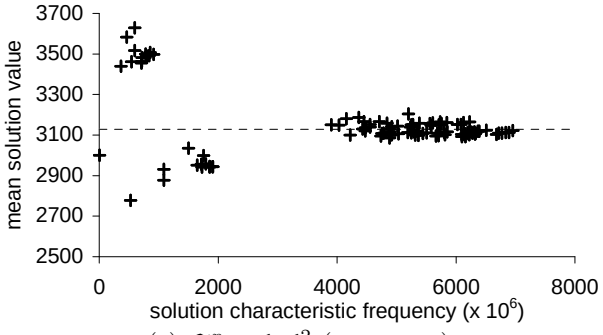
[5] While many GAPs are minimisation problems, the `gap1` and `gap2` problem sets contain maximisation problems.



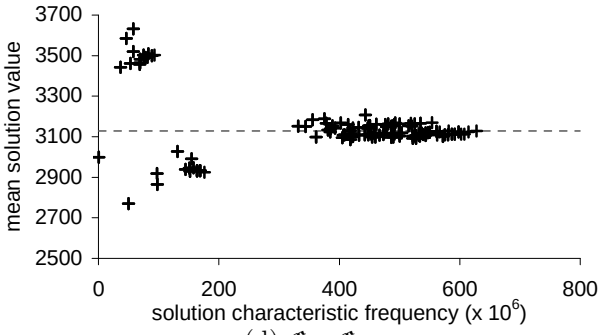
(a) Item weight against frequency of use



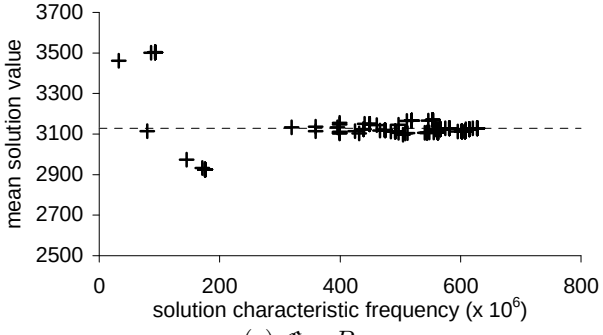
(b) $\mathfrak{C}$



(c) $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)

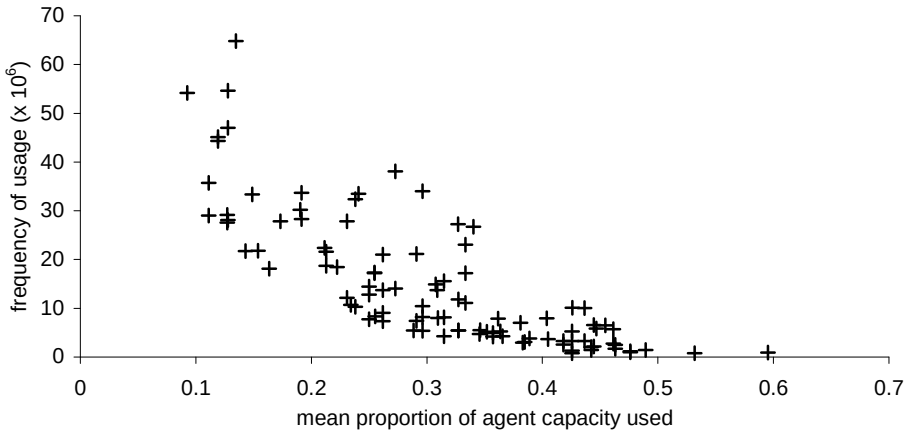


(d) $\mathfrak{C} \times \mathfrak{C}$

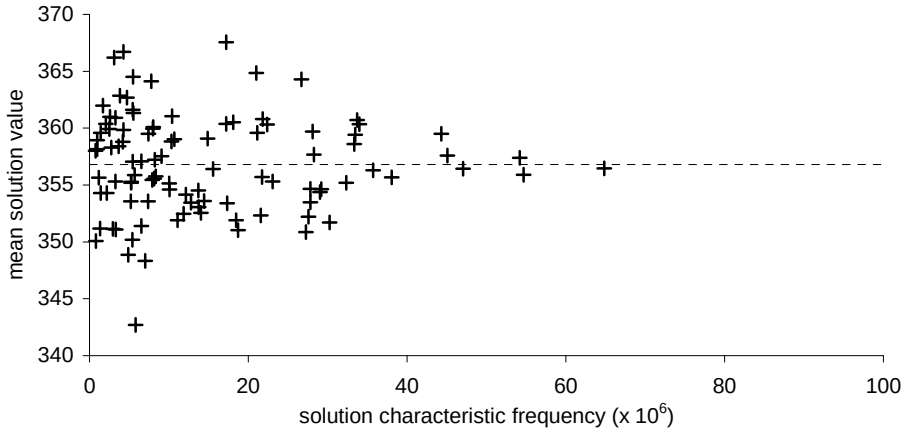


(e) $\mathfrak{C} \times P$

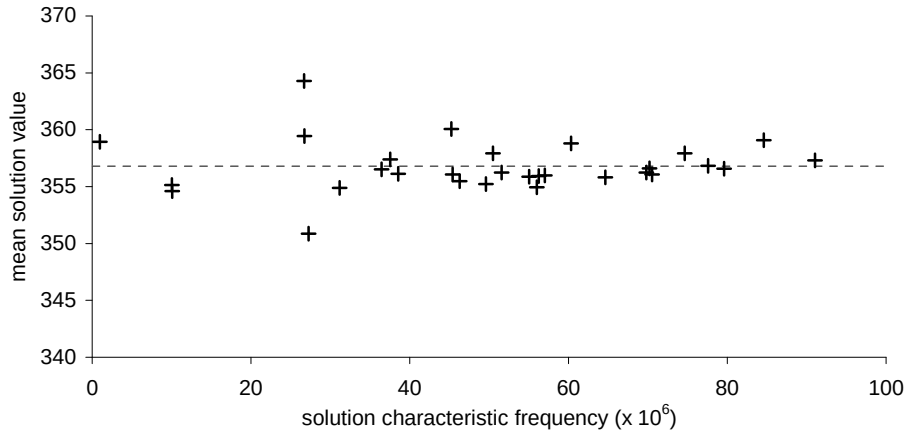
Figure 5.10: Potential bias interaction in the `mknap1-15item` MKP instance. (a) Mean proportion of knapsack capacity used by an item against the mean value of solutions it appears in. (b)–(e) Frequency of use of solution characteristics against mean value of solutions they describe from $\mathfrak{C}$, $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ respectively. The mean value of all sequences is shown as a dashed line.



(a) Agent capacity consumed by assignment against frequency of use



(b) $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$



(c) $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$

Figure 5.11: Potential bias interaction in the `gap2-1` GAP instance. a) Mean proportion of agent capacity used by an assignment against the mean value of solutions it appears in. b) and c) Frequency of use of solution characteristics from $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ respectively against mean value of solutions they describe. The mean value of all sequences is shown as a dashed line.

problems. Enumeration of the construction trees for small instances (`burma14` for the TSP, `tai12a` for the QAP) confirms this supposition. That is, solution characteristics from $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ used with the TSP and from $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ used with the QAP occur with equal frequency within each pheromone type. Although these pheromone representations are free from bias, they would be expected to learn differently as they model different solution characteristics and hence produce differing levels of performance in otherwise equivalent ACO algorithms. The relative performance of these alternative pheromones for these problems is compared in Chapter 7.

5.4 Summary

The effectiveness of a pheromone representation is strongly influenced by the way the solution characteristics it models map onto arcs in the construction tree for a problem. At the lowest level, constructed solution biases will determine the baseline probability of a particular solution characteristic being updated. However, given the influence of these low level biases appears negligible on larger instances, the relative number of sequences each solution characteristic describes may be the ultimate determinant of how often it is reinforced (at least during the early stages of an ACO algorithm when decisions are made more randomly). This latter issue is addressed by Blum's (2004) definition of a CBS, which is a problem–pheromone combination in which every solution characteristic appears the same number of times. If a pheromone representation applied to a problem is not a CBS, that pheromone will exhibit a bias. It has been shown that the presence of an underlying construction bias will prevent any pheromone representation from being a CBS. Additionally, when a representation bias is present, a pheromone may be a CBS yet not be free from solution bias.

In certain problems, notably the JSP and non-OSP GSP instances, the problem's structure may serve to bias a search towards certain kinds of solution. When the solution characteristics modelled by a pheromone representation are associated strongly with one kind of solution or another, that pheromone representation will consequently show a bias towards those kinds of solutions. This is especially the case with $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{rel}}$ applied to the JSP. The definition of a CBS, together with knowledge of the underlying structure and biases in this problem, helps explain why $\mathsf{PH}_{\mathsf{suc}}$ typically performs poorly, while $\mathsf{PH}_{\mathsf{rel}}$ typically performs well.

Unlike the JSP, the MKP and GAP do not appear to have a clearly exploitable structure. In the MKP, solution characteristics associated with the inclusion of items with few resource requirements will be updated most frequently. However, even though items with large resource requirements are often more valuable and so would be expected to be reinforced

less often, the majority of solution characteristics appear in a wide range of solutions, and so will not necessarily bias a search towards good or bad solutions. In the GAP, solution characteristics associated with assignments that utilise less of an agent's capacity will be updated most frequently, but in the absence of a strong relationship between capacity utilisation and value most solution characteristics are not strongly associated with either good or bad solutions.

Even in problems where different pheromone representations show no obviously exploitable bias towards solutions of a certain cost/value, as different pheromones model different aspects of solutions or sequences, they are unlikely to perform equivalently. The next chapter considers the extent to which pheromone representations may be chosen so that the potential for bias is reduced.

Chapter 6

Selecting Pheromone Representations Considering Bias

Chapters 3 and 5 described the potential for unintentional biases to exist in ACO algorithms. While the representation and construction biases that may exist in any constructive algorithm are unlikely to have a noticeable effect on large problem instances, the mechanisms that produce them can serve to bias the various pheromone representations that may be used with a particular problem in different ways. Blum (2004) describes how techniques such as stochastic gradient ascent may be used to control how individual pheromone values are updated to reduce the effects of bias in a given pheromone representation. This chapter deals with the more fundamental question of whether, given a range of pheromone representations that may be used with a problem, there is one representation that is best suited to that problem and which consequently will reduce any bias regardless of how individual pheromone values are updated. Thus it assumes that the utility of a pheromone representation is largely separable from the way in which the information it holds is used.

Section 6.1 reviews the findings of one study into the use of an alternative pheromone update scheme to counteract bias. The remainder of the chapter considers how a pheromone representation may be chosen such that it naturally works against unfavourable biases. Section 6.2 discusses a potentially important feature of effective pheromone representations, which may also serve to reduce some of the effects of bias, i.e., whether or not they allow solutions to be represented more than once. Based on this discussion of the utility of the unique representation of solutions, Section 6.3 presents an algorithm for the derivation of an appropriate pheromone representation based on the characteristics of a problem's objective function.

6.1 Controlling Bias Using the Pheromone Update

Blum (2004) investigated two main ways to counteract undesirable bias effects in ACO algorithms by altering the amount by which pheromone values are reinforced. The first is to use the iteration best solution to update pheromone values, as is done in some ACS and $\mathcal{MM}$AS algorithms, rather than allow all ants to update solutions, as is the case in the original AS. When applying this in an ACO algorithm for the JSP using $\mathsf{PH}_{\mathsf{suc}}$, Blum found the algorithm performed better than using the AS update. However, its performance was still not competitive with that of other pheromone representations. As described in the previous chapter, this problem has an interesting structure that makes $\mathsf{PH}_{\mathsf{suc}}$ particularly susceptible to reinforcing poor solutions.

Blum also discusses the use of stochastic gradient ascent (SGA) (see, e.g., Bertsekas, 1999), noting the relationship between SGA and ACO identified by Meuleau and Dorigo (2002). SGA is a technique for minimising error functions defined over a set of parameters. For use in ACO, the error function is defined as the expected quality of solutions produced given the current pheromone values associated with solution characteristics. While an exact gradient technique would adjust pheromone values so that the likelihood of producing improved solutions always increases, Blum notes that the exact technique requires complete enumeration of the search space. SGA estimates the required change by sampling a number of solutions at each iteration. The actual amount of change is governed by the learning rate, which is a parameter of the approach.

The approach works as follows. At each iteration, the expected quality of solutions is determined by probabilistically constructing a single solution using a modified component selection rule that ensures certain conditions required by the SGA algorithm. Pheromone values are then updated using a modified update rule in which the amount of update depends on the learning rate and the current transition probabilities. Pheromone evaporation is also adjusted by considering the current transition probabilities.[1] It is also noted that using the SGA update allows for greater freedom in the quality function used to assess the relative merits of different solutions, which is a further merit of the technique.

Using this technique, Blum was able to improve the results of an ACO algorithm for the k-cardinality tree problem, producing considerably improved results. The technique is also applied to an ACO for the JSP using $\mathsf{PH}_{\mathsf{suc}}$, producing results comparable with the best performing pheromone when using a standard ACO algorithm and $\mathsf{PH}_{\mathsf{rel}}$. Despite the improved performance achieved by using the SGA update, a number of disadvantages are also identified. First, local search can only be applied to solutions *after* they have been used to update pheromone values, while applying local search to improve solutions before they are used to update pheromone is typically more effective. Second, it was found that

[1] See Blum (2004) for full details of the modifications to the standard ACO approach.

the combined ACO-SGA technique is very sensitive to the chosen learning rate, which depends on the problem instance being solved and the quality function used.

In conclusion, Blum recommends the use of a standard ACO pheromone update in conjunction with an appropriate pheromone model. The latter is the topic of the remainder of this chapter.

6.2 Unique Representation in Pheromone

When applying ACO to the TSP, an intuitive pheromone representation is $\mathfrak{C} \times \mathfrak{C}$, where $\mathfrak{C}$ is the set of cities. This choice is seemingly a good one as it associates pheromone with the solution feature that most directly contributes to cost, i.e., links, and indeed, like many other intuitive pheromone choices, it works quite well in practice (Dorigo and Stützle, 2002). Furthermore, this pheromone representation has an important feature. Each distinct solution to the TSP is represented by exactly one set of links and so, in the pheromone representation, by exactly one set of solution characteristics. (This is not the case, for instance, when using a $\mathfrak{C} \times \mathfrak{C}$ pheromone for a subset problem, as illustrated in Figure 6.1.) Hereafter, pheromone representations with this feature are said to have the *property of unique representation*, expressed by Definition 6.

Definition 6 A pheromone representation C has the *property of unique representation* if each solution is represented by exactly one set of solution characteristics taken from C. □

Note that this definition corresponds with that of an *identity-oriented pheromone representation*, defined in Chapter 4.

Lemma 4 *An identity-oriented pheromone representation necessarily possesses the property of unique representation.* □

Proof The defining characteristic of an identity-oriented pheromone representation, that for each solution s, there is a unique set of solution characteristics drawn from C that describes s, corresponds with Definition 6. ■

There are two criteria for possessing this property. First, each distinct solution must be represented *at least* once, as any excluded solution could be the optimal one. As pheromone representations are derived from characteristics of the solutions or sequences used, which will in any properly designed algorithm allow for any solution to be represented, this criterion is generally trivially met. In some pheromones, distinct solutions may share the same representation, as in Randall's (2002b) ACO algorithm for the static aircraft landing problem. While such pheromones do not exclude any solutions, the sharing of a single representation by multiple solutions may be undesirable on problems where pheromone is

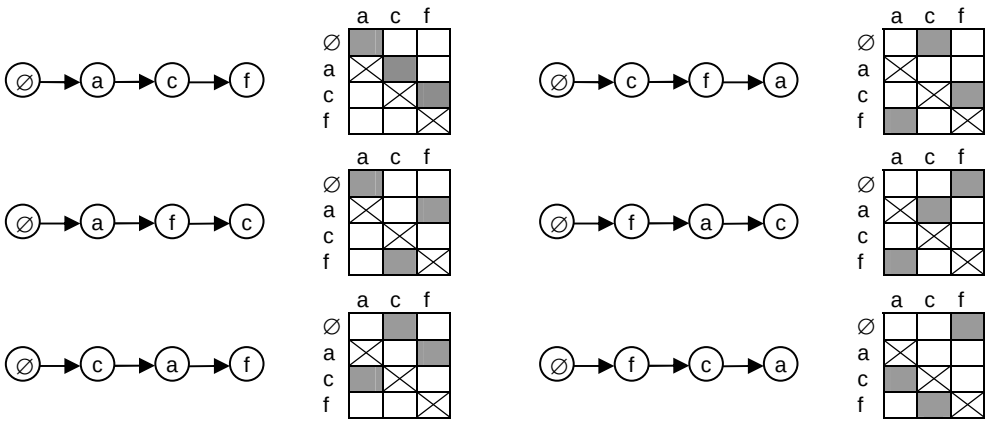


Figure 6.1: A small subset problem (in which the cost is associated with items) using the $\mathfrak{C} \times \mathfrak{C}$ pheromone representation. All possible ant solutions are shown for the subset $\{a, c, f\}$ taken from some larger set. For simplicity, the pheromone matrices shown only have entries for those items in this subset. $\varnothing$ is the artificial start point from which all ants begin. Shaded cells indicate the solution characteristics (pheromone values) corresponding to the adjacent solution. Crossed cells indicate infeasible solution characteristics.

shared between solutions of quite different quality. Such sharing of representations occurs infrequently and is discussed below in Section 6.2.4.

The second criterion for unique representation is that each distinct solution must be represented *no more* than once. If this latter criterion is not met, then ants could construct the same solution while using different sets of solution characteristics (this is illustrated for a subset problem in Figure 6.1). Moreover, pheromones that represent distinct solutions multiple times consequently increase the size of solution space ants search in—even though the same number of distinct solutions are present—which is clearly an undesirable property for search algorithms. Thus, even if a problem has a representation bias, each representation corresponds to the same set of solution characteristics. Notably, in the GBAS, "...to each feasible solution, there corresponds (via Φ^{-1}) at least one walk ...[in the construction graph]" (Gutjahr, 2000, p875). This means that the pheromone it uses, which is associated with the links in these walks, is not guaranteed to have the property of unique representation.

It should be noted that, with any pheromone representation, ants are able to construct the same solution while encountering different sequences and, in some cases, slightly different subsets of the solution characteristics that describe that solution. For instance, in the TSP ants may construct the same solution while starting at different cities, thereby encountering a different sequence of solution characteristics (i.e., links) while never explicitly encountering the link from the last back to the first city which may be explicitly visited by other ants producing the same solution. With higher order pheromone representations, ants may encounter different sets of higher order solution characteristics as the contents

of their respective partial solutions will often be different even if those partial solutions will eventually represent the same solution. Both cases are artefacts of the constructive approach and are not inherently problematic as at each step ants are given the best available estimate of the learned utility of solution components. A problem arises when two or more solution characteristics from a given pheromone representation correspond to a single "true" solution characteristic in terms of a particular problem. For instance, using a $\mathfrak{C} \times P$ pheromone representation for a subset problem means that the solution characteristics $(i, 1), \ldots, (i, n)$ all represent the single solution characteristic "i is present in the solution." Yet with this representation only one of these is used in each ant's solution, spreading pheromone for the important solution characteristic across multiple values. Consequently, each distinct solution also has multiple representations. Indeed, using $\mathfrak{C} \times P$ pheromone for this problem, each solution of n items has $n!$ different representations.

In general, where the solution structure imposed by the constructive process allows for each solution to be represented only once, all alternative pheromones for that structure will have the property of unique representation. Where the solution structure used does allow for distinct solutions to be represented in different ways, pheromones based on how solutions are represented will also represent distinct solutions more than once. Even when alternative pheromones for a problem have the property of unique representation, it does not follow that they will be equally effective, an issue discussed in Section 6.3.

The notion of unique representation is explored in the following sections and the influence of bias is discussed in Sections 6.2.1 and 6.4.

6.2.1 Unique Representation and Bias

Given that any identity-oriented pheromone representation will have the property of unique representation, a pheromone with this property will be susceptible to the same biases as an identity-oriented pheromone. Therefore, based on Lemmas 2 and 3 given in the preceding chapter, a pheromone with this property applied to a problem with either a representation or construction bias cannot be a CBS and *will* exhibit a bias due to those underlying causes.

However, pheromone representations with the property of unique representation still offer the advantage that each distinct solution is represented by one set of solution characteristics and hence, has one learned value. Furthermore, as discussed in Section 6.3 below, such pheromone representations should be the most appropriate models for different combinatorial problems.

6.2.2 Unique Representation in Higher Order Pheromones

If a given first order pheromone representation has the uniqueness property for a particular problem, then any higher order representations generated from it will also have the uniqueness property on the same problem. This is formalised in Theorem 3. This principle holds for all n^{th} order pheromone representations given solutions of at least n first order solution characteristics, as solutions with less do not possess groups of the required size.

Theorem 3 *Let C be an arbitrary first order pheromone representation. If C has the property of unique representation for a given problem, then solutions represented by at least $n \geq 2$ different pheromone values are also uniquely represented by any higher order representation C^n.* □

PROOF Let C be an arbitrary first order pheromone representation such that C has the property of unique representation. Let $C_s \subset C$ be a set of solution characteristics from C such that C_s corresponds to the solution s. Let $C_s^n = \{(i_1, \ldots, i_n) | i_1, \ldots, i_n \in C_s, i_1 \prec \ldots \prec i_n\}$ be the set of all n-tuples of distinct elements from C_s, where $\prec$ is an arbitrary but fixed order imposed on the elements of C. There is only one such set of n-tuples for each C_s and n. From Definition 3 it can be seen that $C_s^n \subset C^n$. Therefore, each solution has exactly one representation in C^n. Based on Definition 6, C^n has the property of unique representation. ■

Corollary 1 *Let C be an arbitrary first order pheromone representation such that C does not have the property of unique representation for a given problem. Provided that solutions are represented by at least n different pheromone values in C, any higher order representation C^n also does not possess the property of unique representation for that problem.* □

PROOF Let C be an arbitrary first order pheromone representation such that C does not uniquely represent solutions. Choose two sets of solution characteristics $C_s, C'_s \subset C, C_s \neq C'_s$ such that both sets correspond to the same distinct solution s. It follows that there are two sets of higher order solution characteristics $C_s^n, C_s'^n \subset C^n, C_s^n \neq C_s'^n$, such that C_s^n corresponds to C_s and $C_s'^n$ corresponds to C'_s. Hence, there are at least two representations of s in each higher order representation C^n, so none of them possess the property of unique representation. ■

6.2.3 Examples of Unique and Multiple Representation

This section examines alternative pheromone representations for three major COPs to illustrate the importance of the property of unique representation. Note that in each case those pheromone representations without this property are *representation-oriented* (see

Section 4.1.1) because in these problems many distinct sequences may map to a single solution.

Subset Problems (cost associated with items)

Consider a subset problem, such as the MKP, SCP, SPP or KCTP, where cost or profit is associated with the items included in the subset. The first three examples have a representation bias as solutions may be of varying length, while the KCTP requires subsets of a fixed size and hence all solutions are represented the same number of times. All four example problems and most other difficult subset problems have a construction bias and so there does not exist a pheromone representation that can be free from bias. Intuitively, such problems should have a $\mathfrak{C}$ pheromone representation, as the items placed in the subset are important solution characteristics. Indeed, a $\mathfrak{C} \times \mathfrak{C}$ pheromone representation is regarded as inappropriate for these problems (Leguizamón and Michalewicz, 1999). Nevertheless, it makes an interesting case study in the analysis of pheromone representations that represent solutions multiple times.

When applying a $\mathfrak{C} \times \mathfrak{C}$ pheromone representation to these problems, a solution characteristic $(\mathfrak{c}_i, \mathfrak{c}_j)$ represents choosing one item $\mathfrak{c}_j$ after choosing some other item $\mathfrak{c}_i$. It is derived from the graph representation of a subset, where the nodes visited are the items included in the subset and pheromone is associated with the links in that graph. If solutions are represented in this way, with an artificial start node (an item with zero weight and cost) from which all ants begin, each solution of k items is represented $k!$ times, with the pheromone associated with including a given item spread over n separate values, where n is the total number of items (excluding the start node), $k < n$. Figure 6.1 illustrates the use of this pheromone representation for a simple subset problem. As described above, a $\mathfrak{C} \times P$ pheromone creates the same number of extra representations for solutions to these problems.

The intuitive pheromone choice $\mathfrak{C}$ satisfies the requirements of unique representation as any solution may be represented yet each solution is represented by at most one subset of solution characteristics.

Graph Colouring Problem

In the GCP, although solution components represent the assignment of specific colours to nodes, distinct solutions are described by the groups of like-coloured nodes. That is, specific colour information is discarded. Thus, for a given k-colouring of a graph, there are $k! - 1$ other colourings that represent the same solution, produced by swapping the actual colours between colour groups. Clearly then, any pheromone representation that includes specific colour information may represent distinct solutions multiple times. This is the case if using

a $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone, where $\mathfrak{C}_{it}$ is the set of nodes and $\mathfrak{C}_{res}$ is the set of colours. Each of the $k! - 1$ alternative representations of each distinct solution has a different corresponding set of solution characteristics (i.e., representation) in the pheromone. Furthermore, such a pheromone may mislead ants by attracting them to make node–colour assignments that, being characteristics of two different representations of the same distinct solution, produce a poorer solution when combined.

Similar problems occur if either of the second order representations $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; (\mathfrak{C}_{it} \times \mathfrak{C}_{res})^2$ or $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$ is used. The former stores pheromone between all pairs of node–colour assignments while the latter only stores pheromone for like coloured pairs of nodes. While these representations capture interdependencies between adjacent nodes so that poor combinations of node–colour assignments are less likely, each solution is still represented multiple times. For instance, consider the solution characteristic (i, j, k) taken from the latter pheromone, which is equivalent to (i, j, k') if the colours k and k' are swapped. As described in Section 4.1, these representations may be simplified by discarding specific colour information to produce a grouping pheromone for this problem. A $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (same group) pheromone, applied to this problem, does have the property of unique representation, as it directly models colour groups.

If this problem is modelled such that the number of colour conflicts is minimised when using a fixed number of colours, then it is free from both representation and construction biases and hence all the pheromone representations described are CBSs. However, if the problem is modelled with the aim of minimising the number of colours required to create a feasible solution, then this may not be the case.

Permutation Scheduling Problems

As described in preceding chapters, ACO algorithms for problems such as the SMTTP, FSP, JSP, GSP and OSP have employed a number of different pheromone representations. Early ACO algorithms (e.g., Colorni et al., 1994) used a TSP-like $\mathfrak{C} \times \mathfrak{C}$ pheromone representation (where $\mathfrak{C}$ is the set of operations), which has the minor disadvantage that an artificial start node representing an empty schedule must be used in order to learn which operation to place first. More recent algorithms have largely used a $\mathfrak{C} \times P$ pheromone, as this appears better suited to modelling permutations. Both of these are representation-oriented and so learn about permutations rather than precedence relationships between operations. As problems such as the JSP, GSP and OSP have a representation bias, representation-oriented pheromones will accordingly represent some solutions multiple times.

Consider the application of a $\mathfrak{C} \times \mathfrak{C}$ pheromone to the trivial scheduling problem depicted in Figure 6.2. In this example, the completion time of o_3 may be affected by the completion time of o_1 (and vice versa), while o_2 is independent of o_1 and o_3. As a result,

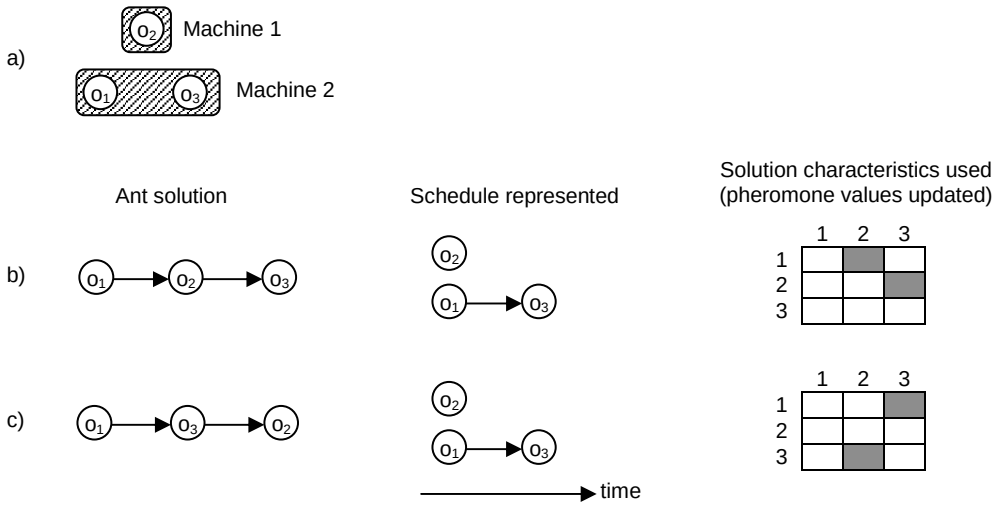


Figure 6.2: A small OSP-style permutation scheduling problem using the $\mathfrak{C} \times \mathfrak{C}$ pheromone representation. The artificial start point $\langle\rangle$ is not shown for brevity. a) o_1 and o_3 must be processed on the same machine, while o_2 requires a different machine. b) An ant's solution, the schedule it represents and the solution characteristics used. c) Another ant's solution, producing the same schedule but using different solution characteristics.

the operation sequence $\langle o_1, o_2, o_3 \rangle$ is equivalent to $\langle o_1, o_3, o_2 \rangle$, even though different solution characteristics from $\mathfrak{C} \times \mathfrak{C}$ are involved in each. As described by Blum and Sampels (2002a), and in Section 5.2, empirical investigation of the $\mathfrak{C} \times \mathfrak{C}$ pheromone representation reveals it has an unfavourable bias such that in more constrained scheduling problems like the JSP, poorer solutions could be reinforced more than better solutions.

The $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) pheromone, developed by Blum and Sampels (2002a) for the GSP, models identifying characteristics of solutions and not of the sequences of operations that represent solutions. Consequently it has the property of unique representation. As reported by Blum and Sampels (2002a) and supported by the empiricial investigation described in Section 5.2, it outperforms the other commonly used pheromone representations for these problems, including $\mathfrak{C} \times P$ with *summation evaluation*.

6.2.4 Shared Representations in Pheromone

When a single set of solution characteristics corresponds to two or more distinct solutions, those solutions may be said to share a representation. In effect, it is the reverse of having multiple representations.[2] However, examination of a wide range of pheromone representation–problem combinations suggests that it is far less common for multiple solutions to share one representation than for a single solution to have multiple representations.

[2]While it would be possible to contrive a pheromone representation that exhibits multiple, shared representations, it is improbable that such pheromones would be developed in the normal course of applying ACO, so such pheromone representations not considered further.

Furthermore, sharing of representations is not necessarily undesirable.

It is possible to contrive unnatural pheromone representations for a variety of problems that exhibit shared representations. For instance, consider a pheromone representation for a subset problem that arbitrarily groups items into pairs with a single pheromone value to indicate if either item in each pair should be included in solutions. Such a representation would make use of a set of virtual components $\mathfrak{C}'$, where $\mathfrak{C} \mapsto \mathfrak{C}'$ and $|\mathfrak{C}'| = |\mathfrak{C}|/2$. Similar pheromone representations may be contrived for other problems by having small groups of solution components where each group has a single pheromone value. Clearly, such representations are inappropriate as they fail to distinguish between different solutions. However, these examples are highly artificial and unlikely to eventuate in the typical application of ACO. Of more interest is the small number of pheromone representations with this property which can appear as a result of plausible design decisions. Two of these are considered here.

Pheromone for the Direct Assignment of Times in Scheduling Problems

Randall's (2002b) ACO algorithm for the static single runway aircraft landing problem uses the pheromone representation $\mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it} \times \mathfrak{C}'_{res}$, where $\mathfrak{C}_{it}$ is the set of planes, $\mathfrak{C}_{res}$ is the set of landing times and $\mathfrak{C}'_{res}$ is the set of time *regions* to which k groups of $\frac{n}{k}$ contiguous timeslots from $\mathfrak{C}_{res}$ are mapped, where n is the number of timeslots available for a particular plane. Aggregating regions of timeslots means that assignments within any given region increase the respective probabilities of all timeslots in that region. Clearly, this sharing of representations in a pheromone model is made possible by the nature of the resources (i.e., timeslots) in this problem. Unlike assignment problems such as the GAP and QAP, where resources are clearly distinct entities, the differences between timeslots are gradual, with nearby times similar in terms of solution cost.

While Randall adopts this approach to deal with planes' often disparate time windows, it may also be advantageous in terms of the ACO algorithm's ability to learn about good solutions. Given there is generally little change in solution cost between assignments to adjacent resources, associating pheromone with individual resources may make it more difficult for the system to learn which resources are good, as an item would have to be assigned the same resource several times before any impact on ant behaviour is observed. To ensure that the best time is chosen, a local search procedure or subdivision of regions could be performed.

Thus, for the single runway aircraft landing problem, modelled as the assignment of timeslots to planes, using a pheromone that shares representations may be a good practical choice.

Car Sequencing Problem

The car sequencing problem is a common problem in the car manufacturing industry (Smith, Palaniswami and Krishnamoorthy, 1996). The aim of the variant considered here is that cars of different models are placed in a production sequence such that the separation penalty between cars of the same model is minimised.[3] Each model has a fixed number of cars. One way this can be modelled is as the allocation of sequence positions to different models of cars.[4] This shares similarities with graph colouring, in that solution cost relates to pairs of items (i.e., sequence positions) assigned to each group (i.e., model), yet it also has features of a typical assignment problem, in that the separation penalty depends on what model a sequence position is assigned to. A pheromone representation that models both of these aspects is developed in Section 6.3.1. However, in order to illustrate potential problems with shared solution representations, only a pheromone for the graph colouring aspect is considered here. A $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$ (same model) pheromone captures how separation penalties are allocated within models. However, while each solution is represented no more than once in this pheromone, several solutions share their representation with other solutions. This is because, unlike graph colouring, the groups in this problem have separate identities, evidenced by the different penalties associated with each model. This pheromone representation is therefore inappropriate as highly different solutions may be represented by the same set of solution characteristics.

Whether or not to share pheromone representations between different solutions seems to be a practical consideration based on the problem in question. The pheromone selection system described in Section 6.3 does not currently support pheromones that share representations between solutions.

6.3 Systematically Determining Appropriate Pheromone Representations

The previous section described the property of unique representation and suggested why it is a desirable property of a pheromone representation. However, in order to devise a system to determine the most appropriate pheromone representation for a problem it is first necessary to consider whether unique representation is sufficient to produce a good

[3] As described in Section 2.4.5, the car sequencing problem may also be modelled as a constraint satisfaction problem, where each station on a production line can support a limited number of cars of the same model in sequence, as in Gottlieb et al. (2003).

[4] In this way the problem is transformed into a GAP instance in which the capacity of each resource (i.e., car model) is the same as the number of cars of that model, and the capacity utilisation for each item (i.e., sequence position) is always 1. Consequently, unlike most GAP instances, there are no infeasible partial solutions.

model of solutions. The following examples consider other factors that may influence the decision of which pheromone representation to use.

A pheromone representation that uniquely represents solutions may not adequately describe solution characteristics that affect solution cost. Consider a subset problem characterised by an objective function where cost is related to some relationship between the pairs of items in the subset, with an objective function of the form $\sum_i \sum_{j \neq i} f(\mathfrak{s}[i], \mathfrak{s}[j])$. The MCP and N-queens problem (NQP) can be formulated in this way. While a $\mathfrak{C}$ pheromone representation has the property of unique representation, the objective function suggests that the impact of including one item in the subset is related to all other items in the subset, which is modelled by the second order pheromone representation $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (co-present). This last pheromone representation is used by Fenet and Solnon (2003) in their ACO algorithm for the MCP.

Socha et al.'s (2002) comparative study of two pheromone representations for the UCTP, the first a standard assignment pheromone ($\mathfrak{C}_{it} \times \mathfrak{C}_{res}$) and the second a grouping pheromone ($\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$ (same group)), found that the first produced better results. While the latter was considered to be more appropriate for timetabling, it may not have been the most appropriate for this particular timetabling problem. A requirement of their algorithm was that only feasible solutions be produced, so this problem is not a classic timetabling problem in which clashes must be minimised. The objective instead was to minimise the number of soft constraint violations, which included a student having a class in the last timeslot of the day, more than two classes in a row, or exactly one class on a day. These soft constraints relate more strongly to the actual times assigned to particular events, which is modelled by the $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone.

Blum and Sampels' (2002a) comparative study of four pheromone representations for the GSP showed that the best performing pheromone modelled those characteristics of solutions that directly affect solution cost. Although this pheromone performs very well due to an advantageous bias (see Section 5.2), it would still model solution characteristics that directly affect cost given an alternative solution construction mechanism that might be less susceptible to bias.

These examples reveal a more general principle concerning problems and the most appropriate choice of pheromone. While multiple solution representations are undesirable, they are actually indications that a pheromone representation fails to adequately model those features of solutions that directly affect solution cost. Ants construct solutions from a set of solution components independent of the pheromone representation. Depending on the problem, each distinct solution may be described by different sets or arrangements of solution components. This is the case in the GCP, where solution components represent node–colour assignments while solutions are only uniquely described in terms of colour

groups. Consider an alternative solution representation based on how various parts of the solution (i.e., solution components or parts of solution components) directly affect solution cost. In effect, this involves defining the decision variables of the problem such that a change in the value of any of them results in a change in the solution represented and hence the cost.[5] As such a set of decision variables ignores how solutions are represented (except where this has an impact on solution cost) it follows that it represents solutions exactly once. In order to adequately model those features of solutions that directly affect solution cost and hence produce a pheromone representation that uniquely represents solutions, solution characteristics must be chosen such that the decision variables they define have this property.

One way to account for a solution component's impact on cost would be to associate pheromone with the edges of the *state graph*. The state graph represents all possible partial solution states. This way, all previous construction decisions are taken into account before considering whether to include a particular solution component. However, such a representation will, for many problems, represent solutions multiple times. Moreover, state graphs for even small problems can be prohibitively large to describe explicitly.

A more feasible approach is to use a pheromone representation such that the solution characteristics it models are those directly related to solution cost. The task of identifying an appropriate pheromone representation then becomes that of identifying how each solution component (on its own or in combination with other solution components) directly affects solution cost. Consideration is restricted to those interactions that *directly affect* solution cost, because to account for all indirect mechanisms through which a solution component may affect solution cost would require complete knowledge of a problem and is therefore infeasible. Considering those interactions that have an identifiable direct impact on solution cost is a practical alternative. This information can be derived from a problem's objective function, or in some cases its objective function and some constraints. Thus, ACO algorithms that have successfully used an intuitive pheromone choice have implicitly been using information contained in the objective function, even though this is rarely recognised. As a first step, only those problems where the objective function alone describes how solution components directly impact on cost are considered.

The objective function for many common COPs consists of a summation over a number of terms. Each of these terms may be considered as a solution characteristic. The problem of deriving an appropriate pheromone representation then becomes that of identifying the nature of these terms. To facilitate this task, a suitable modelling language must be used

[5]It is conceivable using such a representation that two apparently different solutions may have the same objective cost. However, unless they can be shown to be completely equivalent, taking into account factors other than their cost, they should be considered to be separate. For instance, two solutions to the TSP may have the same tour length despite visiting cities in different orders, and would be considered as independent solutions.

that helps reveal how different parts of the solution relate to each other. Hence, a 0-1 integer linear programming formulation would be inappropriate as the true nature of a problem is often difficult to discern given the numerous 0-1 variables and constraints involved. In the following examples, problems have been modelled as sequences of solution components.

The following case study helps to illustrate some of the issues involved in selecting an appropriate pheromone representation.

6.3.1 Case Study: Car Sequencing Problem

Consider the following objective function for the car sequencing problem (CSeqP).

$$Minimise \sum_{i=1}^{M} \sum_{j=1}^{|D(i)|-1} \sum_{k=j+1}^{|D(i)|} P\big(|A(i,k) - A(i,j)|, i\big) \tag{6.1}$$

where $D(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of sequence positions assigned to model i, where k is an integer, $\mathfrak{s}[k] \in \mathfrak{C}_{res} = \{1, \ldots, M\}$ is the model assigned to sequence position $I(k)$ and $I(k)$ is the i^{th} sequence position to be assigned during construction, $A(i,j) \in D(i)$ is the j^{th} sequence position assigned to model i, $P(i,j)$ is the separation penalty for the j^{th} model separated by i places in the sequence, N is the number of cars and M is the number of models.

Note that this model differs from others used with ACO, such as Gottlieb et al. (2003) and Solnon (2000). In both ACO algorithms, which were for a constraint satisfaction variant of the CSeqP, solutions are permutations of cars of different models and so a $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ pheromone is used, where $\mathfrak{C}_{res}$ is the set of car models. The formulation of the problem presented here has been used successfully by Randall (1999) on this variant of the problem.

Given this formulation, this problem has no representation bias as each set of assignments describes a different solution. However, it does have a construction bias because there are a limited number of cars of each model. As each model's set of cars becomes empty during solution construction, which will occur at different points in the construction tree depending on which assignments have been made earlier, the number of alternatives diminishes, creating imbalances in the construction tree. Consequently, pheromone representations for this problem cannot be CBSs.

Given this formulation of the problem it can be difficult to determine an appropriate pheromone representation. Initial analysis of the objective function suggests that solution cost is related to two things: which group (model) a position is assigned to and which pairs of positions are assigned to the same group. These suggest two separate pheromone

representations. Regarding the first, a $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone appears most appropriate, where $\mathfrak{C}_{it}$ is the set of sequence positions and $\mathfrak{C}_{res}$ is the set of models. Given that each solution is completely described by stating which positions cars of a particular model are in, this representation has the property of unique representation. However, it does not capture all aspects of the objective. In Section 6.2.4, a $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; (\mathfrak{C}_{it})^2$ pheromone was considered, as this captures the fact that separation penalties are allocated within models (groups). However, this representation is inappropriate on its own as potentially very different solutions may be represented by the same set of solution characteristics. Clearly, neither of these representations is completely adequate to capture how each part of the solution directly affects solution cost. However, given that a $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ does have the property of unique representation, it is worth considering if a second-order pheromone will suffice.

The second-order pheromone $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; (\mathfrak{C}_{it} \times \mathfrak{C}_{res})^2$ (same group) associates pheromone with pairs of position–model assignments. Based on Theorem 3, this representation also has the property of unique representation. It also correctly captures the two aspects of solution cost in this problem: cost due to assignments is represented by the underlying $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone, while cost related to pairs of sequence positions in the same group is also represented. However, given that the pheromone is to be used to relate only pairs of sequence positions assigned the same model, it can be simplified to $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$. One concern when using higher-order representations is their potential size. Given moderate sized problems involving solution characteristics from $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$, and $|\mathfrak{C}_{it}| \approx |\mathfrak{C}_{res}|$, the storage overhead for a higher-order pheromone representation can become prohibitively large. Thus, provided the number of models in a CSeqP instance is not too large, then the use of a second-order pheromone for the CSeqP is feasible. As an example, the largest problem considered by Randall and Montgomery (2002) consists of 80 cars divided into 4 models, equivalent to a 227 city TSP in terms of pheromone size.

6.3.2 A Decision Algorithm for Pheromone Selection

The algorithm for selecting pheromone representations requires that the problem model has already been parsed and that the nature of entities (i.e., *components*) added at each step is known, even though their impact on solution cost is not. Hence, the algorithm identifies the nature of the solution components or characteristics in the problem. The algorithm has been developed by studying a wide range of commonly occurring COPs to identify recurring themes in the objective function and what these indicate about cost contributors. This is not the only possible algorithm for this task, but appears well suited to those problems examined. The algorithm follows a decision tree (see Figure 6.3) that uses the four questions outlined below. For each question, the possible answers are given,

followed by an explanation of the reason for the question.

Note that $f(1)$ represents a function of one part of the solution while Σ represents an aggregate of many parts of the solution. For example, the SMTTP has an objective in which each operation's contribution to solution cost is related to $\sum_{j=1}^{i} p(\mathfrak{s}[j]) - d(\mathfrak{s}[i])$, where $s[i]$ is the i^{th} operation in the permutation, and $p(k)$ and $d(k)$ represent the processing time and due date of operation k respectively. This corresponds to $f(\Sigma, 1)$. The number partitioning problem (NPP) (Johnson, Aragon, McGeoch and Schevon, 1991) has a single term of the form $\sum_i A(1,i) - \sum_i A(2,i)$, where $A(i,j)$ is the j^{th} number in partition i, which matches $f(\Sigma, \Sigma)$. The algorithm is currently limited to objective functions consisting of no more than two parts. However, a generalised optimisation system developed by Randall and Abramson (2001) makes use of objective function templates (described in Randall (2003b)) in which no more than two parts of a solution are referenced in the objective. Given the wide range of COPs to which that system could be applied, this limit is likely to be sufficient for many of the problems to which ACO is applied.

1. How many parts of the solution are used in each term in the objective?
   Possible answers: $f(1)$, $f(1,1)$, $f(\Sigma, 1)$, $f(\Sigma, \Sigma)$.
   Rationale: To know if cost is related to individual parts of the solution or to some relationship between parts. $f(1)$ restricts the number of feasible pheromone representations while the others require further examination.

2. If each part is related to another part of the solution, how many other parts is it related to?
   Possible answers: *one*, *selected* (and more than one), *all*
   Rationale: To determine the scope of the relationship(s) between parts. The answer *one* restricts the number of feasible pheromone representations while the other two require further examination. *Selected* indicates that each part is related to *some* of the other parts, but not all. For instance, in the JSP the contribution to solution cost of a single operation is related to which other operations that use the same machine have been scheduled before it, which is a subset of all operations since JSPs typically involve multiple machines. In the SMTTP an operation's contribution to solution cost is related to all operations that precede it, but none that succeed it, which also matches the answer *selected*.

3. If each part is related to a selected group of other parts, what identifies them?
   Possible answers: *all preceding/succeeding*, *related preceding/succeeding*, *assigned same group/resource*, *assigned different group/resource*.
   Rationale: To differentiate between problems where the relative order of components is important (first two answers) and those where group assignment is important

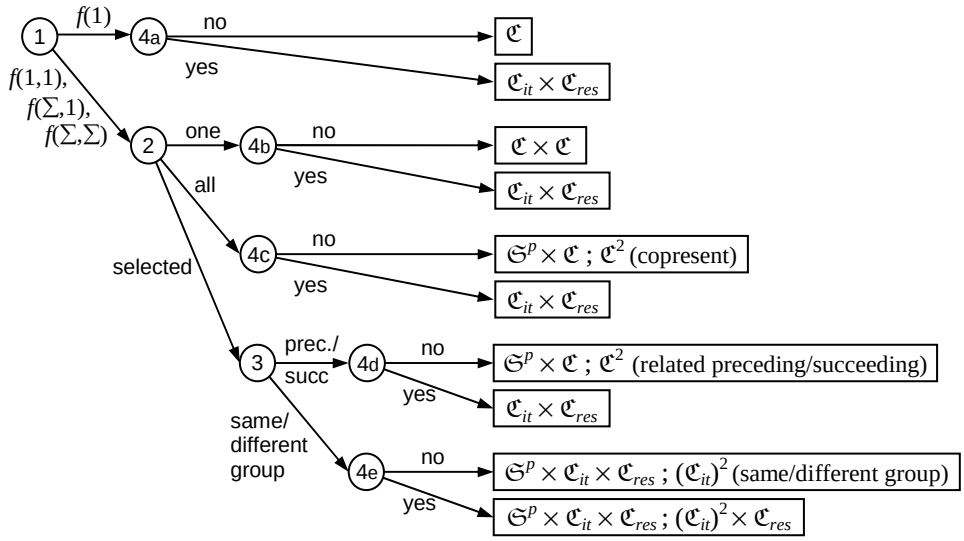


Figure 6.3: Decision tree for selection of pheromone representations. Nodes correspond to the questions outlined in the text. Nodes representing question 4 have been additionally labelled by the letters *a* through *e* to assist in following the example applications of the algorithm given in the text.

(second two answers). In an implementation of the system, the two answers corresponding to each kind of relationship (ordered versus group assignment) would be used to make minor changes to which parts of the pheromone representation would be used in decisions and updated by solutions. For instance, the SMTTP would match *all succeeding* while the JSP would match *related succeeding*.

4. If the part referenced represents an assignment, is the resource assigned used separately in the objective function?
   Possible answers: *yes*, *no* (or solution component is not an assignment).
   Rationale: If solution components represent assignments and each term in the objective is some function of both the item *and* the resource to which it is assigned, then it is likely that it is the assignment that is most important, rather than any other relationship(s) in the problem. The exception to this rule is when dealing with a problem in which group membership is important as well as *which* group items are assigned to, in which case both aspects must be reflected in the pheromone representation.

A simple decision tree is depicted in Figure 6.3. Note that Question 4 appears at the end of most branches and in all but one case overrides any other characteristics of the problem. This is because if the objective is, at some level, a function of individual assignments, then associating pheromone with these will ensure that the representation has the property of unique representation while still representing an important contributor to solution cost.

To illustrate how the algorithm may be applied, consider the following four problems.

Objective functions have been simplified by removing the bounds on summations.

TSP Objective: $\sum_i d(\mathfrak{s}[i], \mathfrak{s}[predecessor(i)])$, where $\mathfrak{s}[i]$ is the i^{th} city in the solution $\mathfrak{s}$, $d(i,j)$ is the distance between cities i and j, and $predecessor(i)$ returns the preceding position to i, which is the last position in the solution if i is the first solution position, $i-1$ otherwise.

- Question 1: $f(1,1)$, goto Question 2;
- Question 2: *one*, goto Question 4b;
- Question 4b: *no*, select $\mathfrak{C} \times \mathfrak{C}$ (adjacent pairs) pheromone.

QAP Objective: $\sum_i \sum_j f\big(I(i), I(j)\big) \cdot d(\mathfrak{s}[i], \mathfrak{s}[j])$, where $I(i)$ is the i^{th} facility assigned, $\mathfrak{s}[i]$ is the location assigned to facility $I(i)$, $f(i,j)$ is the flow between facilities i and j, and $d(k,l)$ is the distance between locations k and l.

- Question 1: $f(1,1)$, goto Question 2;
- Question 2: *all*, goto Question 4c;
- Question 4c: *yes*, select $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone.

GAP Objective: $\sum_{i=1}^{N} C\big(I(i), \mathfrak{s}[i]\big)$, where $I(i)$ is the i^{th} task assigned, $\mathfrak{s}[i]$ is the agent assigned to task $I(i)$, and $C(k,l)$ is the cost of assigning task k to agent l.

- Question 1: $f(1)$, goto Question 4a;
- Question 4a: *yes*, select $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone.

CSeqP A description of this problem appears in Section 6.2.4. In this formulation, sequence positions are allocated to a fixed number of cars within each model.
Objective: $\sum_i \sum_j \sum_k P\big(|A(i,k) - A(i,j)|, i\big)$, where $A(i,j)$ is the j^{th} sequence position assigned to model i, and $P(i,j)$ is the penalty for cars of model j separated by i positions.

- Question 1: $f(1,1)$, goto Question 2;
- Question 2: *selected*, goto Question 3;
- Question 3: *same group*, goto Question 4e;
- Question 4e: *yes*, select $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$ pheromone.

Note that in addition to the decision tree, an implementation of the system must perform some other processing to slightly tailor the chosen pheromone representation to match a particular problem. For instance, with the TSP, the system must be able to recognise that the $predecessor()$ function relates *adjacent* pairs, rather than pairs separated

by some other distance. In problems such as the JSP, the system must be able to determine which other operations are capable of affecting each operation's start time, information obtainable from the problem constraints and objective.

The results from applying the algorithm to a range of COPs are presented in Table 6.1. With the exception of the SMTTP, the suggested pheromone representation is the best known pheromone for each problem to which ACO has been applied in the literature. The suggested pheromone for the SMTTP has not been used with that problem so no comparison can be made.

6.4 Potential to Counteract Bias

Given a problem with no constructed solution biases, any pheromone representation that may be applied to that problem will both represent solutions uniquely and be a CBS. However, it does not follow that every applicable pheromone representation will produce equivalent performance. The review of the ACO literature presented in this thesis supports the claim that modelling solution characteristics that are related to solution cost or value leads to the best results with such problems.

If a problem has a constructed solution bias, then under all practical conditions there does not exist a pheromone representation that will be free from bias. Specifically, if a problem has a construction bias then there does not exist a pheromone representation that is a CBS, and thus a pheromone that represents solutions uniquely will not be free from bias. Furthermore, given what is currently known about pheromone representations with this property, the use of such a pheromone does not entail a reduction in bias. For example, the $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) pheromone used with the JSP represents solutions uniquely and exhibits a strong bias. Fortunately, given the commonality in structure of many JSP instances, this bias typically favours good solutions. However, Blum (2004) describes contrived instances where it performs very badly.

If the nature of the bias in a pheromone representation is unknown or cannot be determined using available analysis techniques, then modelling solution characteristics related to cost offers the following advantages over representation-oriented alternatives. First, as solutions are represented uniquely, the algorithm will learn only one estimate of a solution's quality. Second, there is a clear relationship between the solution characteristics modelled and the impact of their inclusion on solution cost. Both of these suggest that such pheromone representations should learn more effectively which solution characteristics to combine to produce good solutions.

If the nature of the biases in a collection of alternative pheromone representations *is* known, then this knowledge may be used to select the one with the most favourable bias,

even if it does not have the property of unique representation. Thus, it is by considering the existence and, subsequently, nature of biases in pheromone representations that informed decisions concerning the most appropriate pheromone representation may be made.

6.5 Summary

Pheromone representations are generally chosen in an *ad hoc* manner, which has in some instances produced representations that represent solutions multiple times. This increases the effective size of the search space and may potentially mislead the search process as to the true learned value of solutions. In contrast, pheromone representations that represent each solution exactly once may learn more effectively.

Many intuitive pheromone choices have been highly effective in a number of instances, suggesting that deriving pheromone by the systematic analysis of a problem, especially its objective function, may yield improved results. Moreover, deriving a pheromone representation in this way ensures solutions are represented uniquely, while directing learning on those characteristics of solutions that most directly contribute to cost. Given that no pheromone for a problem with a construction bias can be a CBS, such pheromone representations will not be free from bias, yet may counteract some of the effects of bias by supporting more effective learning.

An initial system has been proposed for the selection of pheromone representations that may be applied to a wide range of COPs. In general, this system's predictions correlate with the best-known pheromone representations for problems described in the ACO literature. The next chapter presents the results of empirical investigations into alternative pheromone representations for a number of key problems and compares the results achieved using the system's suggested pheromone against available alternatives for each problem.

Table 6.1: Suggested pheromone representations for common COPs using the new selection system. Problem name abbreviations not introduce previously are as follows: LOP = linear ordering problem, GPP = graph partitioning problem, PAP = processor allocation problem. Other abbreviations used: rel'd = related, prec. = preceding, succ. = succeeding, group = same group, diff. = different group. References are given where ACO has been applied to that problem using the suggested pheromone.

	Decision process					
Problem	Q1	Q2	Q3	Q4	Suggested pheromone	Example
MKP	$f(1)$	—	—	no	$\mathfrak{C}$	Leguizamón and Michalewicz (1999)
SCP	$f(1)$	—	—	no	$\mathfrak{C}$	Fiorenzo Catalamo and Malucelli (2001)
SPP	$f(1)$	—	—	no	$\mathfrak{C}$	
GAP	$f(1)$	—	—	yes	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Lourenço and Serra (2002)
TSP	$f(1,1)$	one	—	no	$\mathfrak{C} \times \mathfrak{C}$	Dorigo et al. (1996)
VRP	$f(1,1)$	one	—	no	$\mathfrak{C} \times \mathfrak{C}$	Bullnheimer et al. (1999a)
NQP	$f(1,1)$	all	—	no	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)	
MCP	$f(1,1)$	all	—	no	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent)	Fenet and Solnon (2003)
QAP	$f(1,1)$	all	—	yes	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Stützle and Dorigo (1999a)
FAP	$f(1,1)$	all	—	yes	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	Maniezzo and Carbonaro (2000)
SMTTP	$f(\Sigma,1)$	all	succ.	no	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (all succ.)	
GSP	$f(\Sigma,1)$	sel'd	succ.	no	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (rel'd succ.)	Blum and Sampels (2002a)
LOP	$f(\Sigma,1)$	all	prec.	no	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (all prec.)	
GCP	$f(1,1)$	sel'd	group	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Costa and Hertz (1997)
CStockP	$f(1,1)$	sel'd	group	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Ducatelle and Levine (2001)
BPP	$f(1,1)$	sel'd	group	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (group)	Ducatelle and Levine (2001)
GPP	$f(1,1)$	sel'd	diff.	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (diff.)	
PAP	$f(1,1)$	sel'd	diff.	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (diff.)	
NPP	$f(\Sigma,\Sigma)$	rel'd	diff.	no	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (diff.)	
CSeqP	$f(1,1)$	sel'd	group	yes	$\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$	

Chapter 7

Pheromone Representation and Assignment Order Comparative Performance

This chapter compares the performance of ACO using alternative pheromone representations for six COPs: the TSP, MKP, GSP, QAP, GAP and CSeqP. The effectiveness of different assignment orders in ACO algorithms for the GAP is also studied. Section 7.1 describes the methodology of the experiments and statistical analyses, while Section 7.2 summarises the analysis results obtained.

7.1 Methodology

Preceding chapters have described the potential for bias in constructive algorithms such as ACO and how this may impact on the performance of different pheromone representations. In particular, Chapter 6 predicts that an appropriate pheromone representation for a problem is one that models solution characteristics that are directly related to solution cost (or value). Given that an underlying construction bias necessarily produces a bias in any pheromone representation used with the problem in question, such pheromone representations will not be free of bias, yet may still offer advantages over alternatives that indirectly model key solution characteristics. The purpose of the investigations described in this chapter is twofold: to test the predictions made in the previous chapter concerning the most appropriate pheromone representation by comparing the relative performance of alternatives; and to examine the performance of different assignment orders in the GAP and identify any impact on pheromone performance. Given that these aims relate to the relative performance of alternative pheromone representations or assignment orders given otherwise identical ACO algorithms, comparisons are not made between the performance

Table 7.1: Problems and pheromone representations studied.

Problem	Entities	Solution form	Applicable pheromone representations
TSP	set of cities $\mathfrak{C}$	permutation of $\mathfrak{c} \in \mathfrak{C}$	$\mathfrak{C} \times \mathfrak{C}$ $\mathfrak{C} \times P$
MKP	set of items $\mathfrak{C}$	sequence of $\mathfrak{c} \in \mathfrak{C}$	$\mathfrak{C}$ $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) $\mathfrak{C} \times \mathfrak{C}$ $\mathfrak{C} \times P$
GSP	set of operations $\mathfrak{C}$	permutation of $\mathfrak{c} \in \mathfrak{C}$	$\mathfrak{C} \times \mathfrak{C}$ $\mathfrak{C} \times P$ $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding)
QAP	set of facilities $\mathfrak{C}_{it}$, set of locations $\mathfrak{C}_{res}$	sequence of $\mathfrak{c} \in \mathfrak{C}_{res}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
GAP	set of tasks $\mathfrak{C}_{it}$, set of agents $\mathfrak{C}_{res}$	sequence of $\mathfrak{c} \in \mathfrak{C}_{res}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
CSeqP	set of sequence positions $\mathfrak{C}_{it}$, set of models $\mathfrak{C}_{res}$	sequence of $\mathfrak{c} \in \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2$ (same model) $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$

of the algorithms described here and those described in the literature.

7.1.1 Problems and Pheromone Representations

Six types of COP were studied, the TSP, MKP, GSP (including JSP and OSP instances), QAP, GAP and CSeqP. These problems were chosen as they are well-studied in the literature, exhibit a range of constructed solution biases and are amenable to solution using a range of different pheromone representations. The QAP, GAP and CSeqP also offer the opportunity to study the effects of different assignment orders on ACO algorithm performance. Table 7.1 describes the key characteristics of the COPs studied and the pheromone representations that were used with each. Note that *solution form* describes the structure of solutions as constructed by an ACO algorithm and so implicitly identifies the solution components from which they are built. The terms *permutation* and *sequence* are used to distinguish between solutions where all solution components must appear exactly once (and where their order is significant) and solutions where this is not the case respectively. With the exception of the algorithms for the TSP, where the initial city is selected randomly, ants start with empty solutions in all algorithms.

Results for each problem are presented in separate sections, with a description of the instances studied given first, followed by a summary of the analyses performed. Summaries of the actual costs/values of solutions produced for each problem are presented in

Appendix D.

7.1.2 Implementation of ACO Algorithms for Studied COPs

In order that differences in the relative performance of alternative pheromones be established independently of the precise way in which pheromone information is updated, two alternative ACO algorithms were implemented: a modified version of ACS and a standard $\mathcal{MM}$AS. Initial testing suggested that the greedy bias of ACS, controlled by the q_0 parameter, often leads to inferior performance than when it is not used. Consequently, this parameter was set to zero in all ACS experiments.

Table 7.2 summarises the parameter values used by the ACO algorithms. Note that the relative influence of heuristic information, controlled by β, was set to different values depending on the problem. These values are summarised in Table 7.3.

Across all the algorithms, the iteration best solution was used to update pheromone. The parameter Q, which controls the amount by which pheromone values are updated, is used only with the TSP, GSP, QAP and CSeqP, which are minimisation problems. As it is typical that $Q = 1$ in $\mathcal{MM}$AS, equations for selecting τ_{min} and τ_{max} were adjusted accordingly. Due to the wide range of costs of solutions to the QAP instances studied the value of Q was set differently for each QAP instance, according to

$$Q = \frac{C_{max}}{1000} \tag{7.1}$$

where $C_{max} = a_{max} \cdot b_{max}$ is a weak upper bound on solution cost where a_{max} and b_{max} are the maximum flow and distance respectively. This results in changes to pheromone values comparable to those in the ACO algorithms for the other problems.

In the two maximisation problems the amount by which pheromone is increased is proportional to $\frac{V}{V_{max}}$, where V is the value of the solution and V_{max} is a weak upper bound on the maximum solution value. For the MKP, V_{max} is given by the sum of all items' profits, while for the GAP it is given by $n \cdot C_{max}$, where n is the number of items and C_{max} the maximum profit from any single assignment.

Algorithms for the TSP, MKP, GSP and GAP were implemented with the capability to use heuristic information. The details and purpose of this information are summarised in Table 7.4. Heuristic information is often related to the change in solution cost (or value) associated with adding a particular solution characteristic. As this is computationally expensive to calculate for the QAP and CSeqP it was not used with those problems. Furthermore, a number of ACO algorithms for the QAP have not used heuristic information (e.g., Stützle, 1997; Taillard and Gambardella, 1997).

Algorithms for the TSP and QAP were implemented with the capability to apply a

Table 7.2: Parameter values used in ACO algorithm implementations.

ACO algorithm	Parameter	Controls	Value
Common	iterations		3000
	m	number of ants	10
	Q	reinforcement amount	100
	γ	global pheromone decay	0.1
ACS	τ_0	initial pheromone level	0.01
	q_0	greedy choice probability	0
	ρ	local pheromone decay	0.015
$\mathcal{MM}$AS	p_{best}	τ_{min}	0.05

Table 7.3: Values of β used with each problem where heuristic information used.

Problem	β
TSP	-2
MKP	2
GSP	-1
GAP	-1

Table 7.4: Heuristic information used by ACO and $\text{ACO}_{\text{undir}}$ algorithms and the characteristics it favours.

Problem	Heuristic measure	Favours
TSP	distance between cities	short distance
MKP	item profit	high profit
GSP	operation processing time	low processing time
GAP	agent capacity consumed by assignment	low capacity utilisation

local search procedure to solutions produced. The local search procedure for the TSP works as follows. Every solution obtainable by inverting sections (of lengths between 2 and $n-1$) of the original solution is examined and the best solution used to replace the original. This process is repeated until a local optimum is found. The local search procedure used with the QAP works as follows. Every solution obtainable by swapping single pairs of assignments is examined and the best solution used to replace the original. As with the TSP, this process is repeated until a local optimum is found. Both procedures are consequently very powerful, but also computationally very costly and hence were only used with TSP instances up to size 100 and QAP instances up to size 35.

In addition to ACS and $\mathcal{MM}$AS ACO algorithms, otherwise equivalent implementations of $\text{ACO}_{\text{undir}}$ and ACO_{heur} (i.e., $\text{ACO}_{\text{undir}}$ biased by heuristic information) were examined for each problem.

All ACO algorithms were implemented in the C language and experiments were executed on a 2.6GHz Pentium 4 with 512Mb of RAM, running Red Hat Linux 8.0.

7.1.3 Variables and Analysis Techniques Used

Each combination of pheromone representation (including no pheromone, $\text{ACO}_{\text{undir}}$), assignment order (where supported), and the use or not of heuristic information or local search (where supported) was tested across 10 random seeds. The output from each experiment is a frequency distribution of the costs (or values) of solutions produced during the algorithm's run, expressed as the relative percentage deviation (RPD) from the optimal solution cost or value. RPD is calculated according to

$$RPD = \frac{|C - C_{best}|}{C_{best}} \tag{7.2}$$

where C is the solution's cost/value and C_{best} is the optimal (or best known) solution cost/value for the problem in question. Treating solutions' costs/values in this way allows minimisation and maximisation problems to be analysed uniformly and for results from different instances of the same problem type of problem to be combined. As the cost/value distributions from each run are non-normal, the data were summarised using the minimum and median RPD. Comparisons between different pheromone representations and assignment orders were conducted using these two measures.

As the distribution of minimum and median RPD values within each problem type is also non-normal, nonparametric statistical tests were used to perform comparisons. Comparisons between pairs of pheromone representations (or between ACO and $\text{ACO}_{\text{undir}}$) were performed within each combination of the other independent variables (i.e., use or not of heuristic information or local search and assignment order used with the QAP, GAP and

CSeqP). Similarly, comparisons of assignment orders for the GAP were performed within each combination of the other independent variables and were conducted on the raw proportion of feasible solutions produced.

In the remainder of this chapter a pheromone representation, such as $\mathfrak{C} \times \mathfrak{C}$ is used to refer to ACO using that pheromone representation.

Example Comparison Table

Comparisons between alternative pheromones (and $\text{ACO}_{\text{undir}}$) and between alternative assignment orders for the GAP are presented in tabular form as shown in Table 7.5 below. This is a contrived example comparing the minimum RPD results of $\text{ACO}_{\text{undir}}$ with $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$, and is read as follows. To determine how the performance of $\text{ACO}_{\text{undir}}$ compares against $\mathfrak{C} \times \mathfrak{C}$, locate $\text{ACO}_{\text{undir}}$ in the left-most column and then within that row locate the cell in the column under $\mathfrak{C} \times \mathfrak{C}$. In this instance, the $>$ symbol indicates that $\text{ACO}_{\text{undir}}$ produced higher values for minimum RPD than $\mathfrak{C} \times \mathfrak{C}$. The "($\alpha$ =1%)" below indicates that this result is statistically signifcant at the 1% level. It does not indicate the magnitude of the difference. Each comparison is presented once, so if the difference between $\mathfrak{C} \times P$ and $\mathfrak{C} \times \mathfrak{C}$ were sought, it is $\mathfrak{C} \times \mathfrak{C}$ that must be located in the left-most column.

If a difference exists between two alternatives but is not statistically significant, then no significance level is shown below the direction indicator. The $=$ symbol is used if the probability that the observed difference is due to chance is above 85%, and hence it is unlikely there exists an actual difference in relative performance.

As comparisons of solution cost/value are performed on the RPD, $A < B$ indicates that A produces better solutions than B, while $A > B$ indicates the reverse. In comparisons of the proportion of feasible solutions produced to the GAP instances, $A > B$ indicates that A produces more feasible solutions than B and hence A may be considered to perform better than B in this respect.

Table 7.5: Example comparison table. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test. Comparisons are read by locating the first pheromone A in the left-most column and then taking the difference direction indicator, such as $<$, from the the column corresponding to the second pheromone B, indicating the result $A < B$.

	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$<$ (α =10%)
$\mathfrak{C} \times \mathfrak{C}$		$<$

7.2 Results

7.2.1 TSP

The TSP instances studied are summarised in Table 7.6. All of these are available at the TSPLIB (Reinelt, 2004).

Table 7.6: TSP instances.

Instance	Size	Optimal cost
gr24	24	1272
hk48	48	11461
eil51	51	426
st70	70	675
eil76	76	538
kroA100	100	21282
d198	198	15780
lin318	318	42029
pcb442	442	500778
att532	532	27686

Pheromone Representation Comparative Performance

The two pheromones used with the TSP were $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$. A Kruskal-Wallis test comparing $\text{ACO}_{\text{undir}}$, $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ shows statistically significant differences between the three. Table 7.7 shows the results of Mann-Whitney tests comparing pairs of algorithms. Although both minimum and median RPD were compared, the results are almost entirely identical and hence only one table of comparisons is shown. The order of performance of the alternatives was fairly constant across ACS and $\mathcal{MM}$AS, as well as when heuristic information and local search are used. The only exceptions are:

- When neither heuristic information nor local search is used, the difference between $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ is not statistically significant, although the direction of difference is the same.
- When heuristic information is used, but local search is not used, there is no statistically significant difference between the minimum RPD of $\text{ACO}_{\text{undir}}$ and $\mathfrak{C} \times P$.

It should be noted that the performance of all three alternatives is significantly improved by the use of heuristic information.

As the constructive algorithm used in these experiments chooses the first city randomly, it was considered possible that this might unfairly disadvantage $\mathfrak{C} \times P$, which learns the

Table 7.7: Pairwise comparisons of minimum RPD achieved by $\text{ACO}_{\text{undir}}$ and ACO with different pheromone representations for the TSP. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$>$ (α =1%)
$\mathfrak{C} \times \mathfrak{C}$		$<$ (α =1%)

absolute position of cities in the permutation. However, tests involving $\mathfrak{C} \times P$ and a fixed initial city produced poorer results than the original.

Overall, $\text{ACO}_{\text{undir}}$ (here representing both itself and its heuristically guided counterpart ACO_{heur}), $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ may be ranked in terms of solution cost as

$$\mathfrak{C} \times \mathfrak{C} \quad \prec \quad \mathfrak{C} \times P \quad \prec \quad \text{ACO}_{\text{undir}}$$

where $A \prec B$ means that, in general, A produces better solutions than B.

7.2.2 MKP

The MKP instances studied are summarised in Table 7.8. Instances from the `mknap1` problem set are named according to `mknap1-`n`item`, where n is the number of items in the problem. The `mknapcb1` problem set, developed and studied by Chu and Beasley (1998b), contains 30 instances, the first 10 with a tightness ratio of 0.25, the next 10 with a tightness ratio of 0.5 and the remainder with a tightness ratio of 0.75.[1] The instances used in this study are named according to `mknapcb1-`t`-`i, where t is the tightness ratio of the instance and i is the index of that instance within the list of instances with that tightness ratio.

Pheromone Representation Comparative Performance

The four pheromone representations examined with the MKP were $\mathfrak{C}$, $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$. The last two are representation-oriented pheromones that may be used as solutions are constructed as sequences of the items included. The second is a second order version of $\mathfrak{C}$, which has the following details when placed in the framework introduced in Section 4.2. Given a set of solution characteristics $C = \mathfrak{C}$, the pheromone

[1] The tightness ratio is a measure of the tightness of knapsack capacity constraints, calculated according to $b_i / \sum_{i=1}^{n} r_{ij}$ where b_i is the capacity of knapsack i, r_{ij} is the amount of knapsack i's capacity required by item j and n is the number of items (Chu and Beasley, 1998b).

Table 7.8: MKP instances.

Instance	Items	Knapsacks	Optimal value	Best known value
mknap1-6item	6	10	3800	
mknap1-10item	10	10	8706.1	
mknap1-15item	15	10	4015	
mknap1-20item	20	10	6120	
mknap1-28item	28	10	12400	
mknap1-39item	39	5	10618	
mknap1-50item	50	5	16537	
mknapcb1-0.25-1	100	5		24381
mknapcb1-0.25-2	100	5		24274
mknapcb1-0.25-3	100	5		23551
mknapcb1-0.5-1	100	5		42757
mknapcb1-0.5-2	100	5		42545
mknapcb1-0.5-3	100	5		41968

associated with adding the item $i \in C$ to the partial solution $\mathfrak{s}^p$ is

$$\tau(\mathfrak{s}^p, i) = \begin{cases} mean(\mathfrak{s}^p, i, \tau_2) & \text{if } |C_i| > 0 \\ \tau_1 & \text{otherwise} \end{cases} \tag{7.3}$$

where τ_2 is the pheromone representation describing pairs of items being copresent in a solution, $\tau_1(i) = \tau_0 \ \forall i \in \mathfrak{C}$ (i.e., the initial component is selected randomly, which encourages exploration), $C_i = \{i' \in \mathfrak{s}^p\}$, and the function $mean$ is defined as

$$mean(\mathfrak{s}^p, i, \tau_2) = \frac{\sum_{i' \in C_i} \tau_2(i, i')}{|C_i|} \tag{7.4}$$

where $\tau_2(i, i')$ is the learned utility of including both item i and item i' in the same solution. The mean of relevant pheromone values is used in preference to the sum so that when heuristic information is used the influence of pheromone remains the same regardless of the size of the solution.

Across the four combinations of ACO algorithm (ACS or $\mathcal{MM}$AS) and heuristic versus no heuristic information the following relationships were found to be consistent:

- ACO with any pheromone outperforms $\text{ACO}_{\text{undir}}$ and ACO_{heur}.
- All other pheromones outperform $\mathfrak{C} \times \mathfrak{C}$.
- $\mathfrak{C}$ pheromone outperforms $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent) pheromone.

The performance of all algorithms was improved by the use of heuristic information.

Table 7.9: Pairwise comparisons of minimum RPD achieved by $\text{ACO}_{\text{undir}}$ and ACO with different pheromone representations for the MKP, under ACS and $\mathcal{MM}$AS when using heuristic information. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	ACS				$\mathcal{MM}$AS			
	$\mathfrak{C}$	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$	$\mathfrak{C}$	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$
ACO_{heur}	> (α =1%)	> (α =1%)	> (α =1%)	> (α =1%)	> (α =1%)	> (α =1%)	> (α =1%)	> (α =1%)
$\mathfrak{C}$		< (α =10%)	< (α =1%)	< (α =5%)		<	< (α =10%)	> (α =5%)
$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$			< (α =1%)	< (α =10%)			< (α =5%)	> (α =1%)
$\mathfrak{C} \times \mathfrak{C}$				> (α =10%)				> (α =1%)

However, the results also showed some sensitivity to the ACO algorithm used and to whether or not heuristic information was used. In particular, when heuristic information is not used, the most effective pheromone representation is actually $\mathfrak{C} \times P$, followed by $\mathfrak{C}$ and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), which performed similarly. When heuristic information is used with ACS, $\mathfrak{C}$ outperforms $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (copresent), which outperforms $\mathfrak{C} \times P$. However, the best results (for all pheromones) were achieved when using $\mathcal{MM}$AS with heuristic information, where $\mathfrak{C} \times P$ pheromone produces the best results. Table 7.9 summarises the comparisons of minimum RPD for both ACS and $\mathcal{MM}$AS using heuristic information. As the single pheromone value from $\mathfrak{C}$ associated with including an item is, in effect, split across multiple pheromone values when using $\mathfrak{C} \times P$, it was considered possible that the algorithm converges too quickly when using $\mathfrak{C}$. However, experiments in which the amount by which pheromone values were updated was reduced produced inconsistent improvement when using $\mathfrak{C}$.

In these experiments the following ranking of alternative pheromones and $\text{ACO}_{\text{undir}}$ (or ACO_{heur}) in terms of solution value appears to hold when either heuristic information is not used or the ACO algorithm is $\mathcal{MM}$AS:

$$\mathfrak{C} \times P \quad \succ \quad \mathfrak{C} \quad \succ \quad \mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2 \text{ (copresent)} \quad \succ \quad \mathfrak{C} \times \mathfrak{C} \quad \succ \quad \text{ACO}_{\text{undir}}$$

where $A \succ B$ means that, in general, A produces better solutions than B, while when ACS is used with heuristic information:

$$\mathfrak{C} \quad \succ \quad \mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2 \text{ (copresent)} \quad \succ \quad \mathfrak{C} \times P \quad \succ \quad \mathfrak{C} \times \mathfrak{C} \quad \succ \quad \text{ACO}_{\text{undir}}$$

The often superior performance of $\mathfrak{C} \times P$, a pheromone that has never been considered for use with the MKP and which intuitively appears inappropriate, requires further investigation.

7.2.3 GSP

The GSP instances studied are summarised in Table 7.10. These instances are from a data set used by Blum (2004). The `whizzkids97-gsp` instance is a true GSP instance, developed for a timetabling competition (Blum, 2004), while `whizzkids97-jsp` and `whizzkids97-osp` are JSP and OSP versions of the original respectively. All other instances are named according to `name-`g, where `name` is the name of the original benchmark JSP instance and g is the group size used to create a GSP instance. Hence, `ft10-1` is the original `ft10` JSP instance, `ft10-10` is an OSP instance derived from that, and `ft10-5` is an intermediate GSP instance.

Best known costs are those reported by Blum (2004), with the exception of the JSP and OSP `whizzkids97` instances for which no results could be found in the literature. The best known costs used for these two instances are those obtained in this study.

Table 7.10: GSP instances.

Instance	Operations	Jobs	Machines	Optimal cost	Best known cost
ft10-1	100	10	10	930	
ft10-5	100	10	10		748
ft10-10	100	10	10		655
la38-1	225	15	15	1196	
la38-8	225	15	15		954
la38-15	225	15	15		943
whizzkids97-jsp	197	20	15		601
whizzkids97-gsp	197	20	15	469	
whizzkids97-osp	197	20	15		379

Pheromone Representation Comparative Performance

The pheromone representations used with the GSP were $\mathfrak{C} \times \mathfrak{C}$, $\mathfrak{C} \times P$ and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) (referred to as $\mathsf{PH}_{\mathsf{suc}}$, $\mathsf{PH}_{\mathsf{pos}}$ and $\mathsf{PH}_{\mathsf{rel}}$ in Chapter 5). Considering all problem instances, $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) clearly outperforms all other approaches, regardless of the ACO algorithm used and whether or not heuristic information is used. $\mathfrak{C} \times P$ is next best, and produces almost equivalent results when OSP instances are considered separately. Table 7.11 shows the results of Mann-Whitney tests comparing alternative approaches when heuristic information is not used. These results were the same for ACS and $\mathcal{MM}$AS.

Considering results for each problem type, JSP, GSP and OSP, $\text{ACO}_{\text{undir}}$ (and ACO_{heur}) outperforms $\mathfrak{C} \times \mathfrak{C}$ on both the JSP and GSP. However, $\mathfrak{C} \times \mathfrak{C}$ outperforms $\text{ACO}_{\text{undir}}$ (and ACO_{heur}) on OSP instances. With the exception of GSP (i.e., not JSP or OSP) instances

Table 7.11: Pairwise comparisons of minimum RPD achieved by $\text{ACO}_{\text{undir}}$ and ACO using different pheromone representations for the JSP, GSP and OSP when heuristic information is not used. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$
$\text{ACO}_{\text{undir}}$	$>$	$<$	$>$
	(α =1%)		(α =1%)
$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$		$<$	$<$
		(α =1%)	(α =1%)
$\mathfrak{C} \times \mathfrak{C}$			$>$
			(α =1%)

Table 7.12: Pairwise comparisons of minimum RPD achieved by $\text{ACO}_{\text{undir}}$ and ACO with different pheromone representations for the OSP when heuristic information is not used. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times \mathfrak{C}$	$\mathfrak{C} \times P$
ACO_{heur}	$>$	$>$	$>$
	(α =1%)	(α =1%)	(α =1%)
$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$		$<$	$<$
		(α =1%)	
$\mathfrak{C} \times \mathfrak{C}$			$>$
			(α =1%)

when not using heuristic information, all of these differences are statistically significant at the 1% level. Table 7.12 shows the results of Mann-Whitney tests comparing alternative approaches to the OSP instances when heuristic information is not used. Note that when heuristic information is used the difference between $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) and $\mathfrak{C} \times P$ becomes statistically significant.

Considering the JSP, GSP and OSP instances as a whole, or within JSP and GSP instances, the following ranking of alternative pheromones and $\text{ACO}_{\text{undir}}$ (or ACO_{heur}) holds:

$$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2 \text{ (related succeeding)} \quad \prec \quad \mathfrak{C} \times P \quad \prec \quad \text{ACO}_{\text{undir}} \quad \prec \quad \mathfrak{C} \times \mathfrak{C}$$

where $A \prec B$ means that, in general, A produces better solutions than B. On OSP instances, the relative order $\mathfrak{C} \times \mathfrak{C}$ and $\text{ACO}_{\text{undir}}$ (and ACO_{heur}) is reversed.

7.2.4 QAP

The QAP instances studied are summarised in Table 7.13. They are all available at the QAPLIB (Burkard, Çela, Karisch and Rendl, 2004).

Table 7.13: QAP instances.

Instance	Size	Optimal cost	Best known cost
nug12	12	578	
tai12a	12	224416	
nug15	15	1150	
nug20	20	2570	
tai25a	25	1167256	
nug30	30	6124	
tai35a	35		2422002
ste36a	36	9526	
tho40	40		240516
sko49	49		23386
tai50a	50		4938796
sko56	56		34458
sko64	64		48498

Pheromone Representation Comparative Performance

The two pheromone representations used with the QAP were $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$, where $\mathfrak{C}_{it}$ is the set of facilities and $\mathfrak{C}_{res}$ is the set of locations. The first pheromone is typical of those used for assignment problems, while the latter is an alternative representation-oriented pheromone that may be used because solutions are permutations of elements from $\mathfrak{C}_{res}$. The elements of $\mathfrak{C}_{res}$ are augmented by an element corresponding to the empty solution so that it can learn which location to assign to the first facility. To assess the sensitivity of the algorithm to the assignment order used, three assignment orders were used:

Static, fixed order (SFO) A static assignment order where facilities are assigned in the same order as they appear in the instance description.

Dynamic, randomised order (DRO) The next facility to assign is chosen randomly according to a uniform distribution.

High flow first (HF) A static order in which facilities are assigned in non-increasing order of their total flow requirements.

As $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ requires a static assignment order to be sensibly applied, comparisons between the two pheromones were carried out only within results for SFO and HF assignment orders. Comparisons between $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ were conducted for all assignment orders.

Table 7.14 shows the results of Mann-Whitney tests comparing alternative approaches when local search is not used. These results vary little between ACS or $\mathcal{MM}$AS, with the

Table 7.14: Pairwise comparisons of minimum RPD achieved by $\text{ACO}_{\text{undir}}$ and ACO with different pheromone representations for the QAP, under ACS and $\mathcal{MM}$AS when local search is not used. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

Using SFO

	ACS		$\mathcal{MM}$AS	
	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$>$ (α =10%)	$>$ (α =1%)	$>$ (α =1%)
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$		$<$ (α =1%)		$<$ (α =1%)

Using HF

	ACS		$\mathcal{MM}$AS	
	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$>$ (α =1%)	$>$ (α =1%)	$>$ (α =5%)
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$		$<$ (α =1%)		$<$ (α =1%)

exception that the difference between $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ is not statistically significant under $\mathcal{MM}$AS. Using DRO, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ consistently produces better solutions than $\text{ACO}_{\text{undir}}$ when local search is not used. Thus, when local search is not used, $\text{ACO}_{\text{undir}}$, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ may be ranking in terms of solution cost as

$$\mathfrak{C}_{it} \times \mathfrak{C}_{res} \quad \prec \quad \mathfrak{C}_{res} \times \mathfrak{C}_{res} \quad \prec \quad \text{ACO}_{\text{undir}}$$

where $A \prec B$ means that, in general, A produces better solutions than B.

When local search is used, $\text{ACO}_{\text{undir}}$, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ were able to find the optimal solution to the `nug12`, `tai12a`, `nug15` and `nug20` instances in every case, and to the `nug30` instance in many cases. On the `nug30` instance, $\text{ACO}_{\text{undir}}$ was able to find the optimal solution more often than either $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ or $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$. On two of the largest instances with which local search was used, `tai25a` and `tai35a`, all approaches were able to find good solutions (within 2% of the optimal), and in some cases were able to find optimal solutions. However, in most cases $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ found better solutions than $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ and found the optimal solution on a greater number of runs. Furthermore, comparing the median RPD results for the three reveals that $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ finds proportionally more good quality solutions than $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$, which in turn produces proportionally more good quality solutions than $\text{ACO}_{\text{undir}}$. The ability of $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ to converge on good quality solutions when local search is used suggests that on larger instances it may perform better than $\text{ACO}_{\text{undir}}$ with local search.

7.2.5 GAP

The GAP instances studied are summarised in Table 7.15. They are available at the OR-Library (Beasley, 2005). The names used here are derived according to `[problem_set]`-i, where `[problem_set]` is the name of the set of instances and i is the index of the instance within that set.

Table 7.15: GAP instances.

Instance	Tasks	Agents	Optimal value
gap1-1	15	5	336
gap1-2	15	5	327
gap1-3	15	5	339
gap1-4	15	5	341
gap1-5	15	5	326
gap2-1	20	5	434
gap2-2	20	5	436
gap2-3	20	5	420
gap2-4	20	5	419
gap2-5	20	5	428
gap3-1	25	5	580
gap4-1	30	5	656
gap5-1	24	8	563
gap6-1	32	8	761
gap7-1	40	8	942
gap8-1	48	8	1133
gap9-1	30	10	709
gap10-1	40	10	958
gap11-1	50	10	1139
gap12-1	60	10	1451

GAP Assignment Orders Studied

The assignment orders studied in Section 3.3.1 were used with each combination of ACO algorithm and pheromone representation for the GAP. The *static least constrained* assignment order was not studied as it was proposed as an example of a poor assignment order and has been demonstrated to be so. Below is a list of the assignment orders used (full descriptions are given in Section 3.3.1):

- Static, fixed order (SFO)
- Dynamic, randomised order (DRO)
- Static most constrained (SMC)

- Dynamic most constrained (DMC)
- Dynamic candidate set (DCS)
- Dynamic, probabilistic using a static constrainedness measure (DPS)
- Dynamic, probabilistic using a dynamic constrainedness measure (DPD)

Initial analysis was performed on each of the last five assignment orders to determine which of the two constrainedness measures described in Section 3.3.1 produces a higher proportion of feasible solutions. A statistically significant difference was only found between DMCa and DMCp (i.e., DMC using either the absolute or proportional measure of constrainedness), with DMCa producing more feasible solutions. All other assignment orders performed slightly better using the proportional constrainedness measure. Subsequent comparisons were conducted for only the better performing variant of each assignment order.

The next section reports on the relative performance of different pheromones (and $\text{ACO}_{\text{undir}}$) across assignment orders, while the subsequent section reports on the relative performance of different assignment orders when using either $\text{ACO}_{\text{undir}}$ or the best performing pheromone for this problem.

GAP Pheromone Representation Comparative Performance

The two pheromone representations used with the GAP were $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$, where $\mathfrak{C}_{it}$ is the set of tasks and $\mathfrak{C}_{res}$ is the set of agents. As with the QAP, the latter is an alternative representation-oriented pheromone that may be used when solutions are permutations of elements from $\mathfrak{C}_{res}$. The elements of $\mathfrak{C}_{res}$ are augmented by an element corresponding to the empty solution so that it can learn which agent to assign to the first task. To assess the sensitivity of the algorithm to the assignment order used, comparisons where conducted within each of the assignment orders described above. As $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ requires a static assignment order to be sensibly applied, comparisons between the two pheromones were carried out only within results for SFO and SMC. Comparisons between $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ were conducted for all assignment orders.

Table 7.16 shows the results of Mann-Whitney tests comparing alternative approaches when heuristic information is not used. These results were the same for both SFO and SMC and when using ACS or $\mathcal{MM}$AS, with the following exceptions:

- When using SFO and heuristic information, ACO_{heur} performs equivalently to $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ under $\mathcal{MM}$AS in terms of solution value.
- When using SMC and heuristic information with ACS, the difference in minimum RPD between $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ is significant at the 5% level.

Table 7.16: Pairwise comparisons of pheromone representations for the GAP when using either SFO or SMC assignment orders when heuristic information is not used. The direction of any difference between minimum RPD and the proportion of feasible solutions produced (labelled "feasible solutions") is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	minimum RPD		feasible solutions	
	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$>$	$<$ (α =1%)	$<$ (α =1%)
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$		$<$ (α =1%)		$>$ (α =1%)

Results for comparisons involving median RPD were the same, with the exception that the difference between $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ was found to be statistically significant when using heuristic information.

Comparisons between $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ with the dynamic assignment orders found that $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ produces better solutions and more feasible solutions than $\text{ACO}_{\text{undir}}$ regardless of the assignment order, ACO algorithm or whether heuristic information is used or not. All results are statistically significant at the 1% level.

Across assignment orders, $\text{ACO}_{\text{undir}}$ (also representing ACO_{heur}), $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ may be ranked in terms of solution value as

$$\mathfrak{C}_{it} \times \mathfrak{C}_{res} \quad \succ \quad \mathfrak{C}_{res} \times \mathfrak{C}_{res} \quad \succ \quad \text{ACO}_{\text{undir}}$$

where $A \succ B$ means that, in general, A produces better solutions than B. In terms of the proportion of feasible solutions produced, the three alternatives may be ranked in the same order, where $A \succ B$ is interpreted as A produces more feasible solutions than B. This last result is to be expected given that both $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ are only reinforced by feasible solutions, and so are more likely to make assignments that will lead to more feasible solutions being produced.

GAP Assignment Order Comparative Performance

The performance of different assignment orders was compared within results for $\text{ACO}_{\text{undir}}$ and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$, in terms of both the proportion of feasible solutions produced and the value of those solutions.

Based on a Kruskal-Wallis test and subsequent pairwise comparisons using the Mann-Whitney test of the proportion of feasible solutions produced by $\text{ACO}_{\text{undir}}$ under different assignment orders, the following ranking can be made:

$$\text{SFO} \prec \text{DRO} \preceq \text{DCS} \prec \text{DPD} \preceq \text{DPS} \prec \text{SMC} \preceq \text{DMC}$$

where $A \prec B$ indicates there is strong statistical evidence that A typically produces fewer feasible solutions than B, while $A \preceq B$ indicates that while A often produces fewer feasible solutions than B, the result is not statistically significant. This is similar to the results described in Section 3.3.1, except that the differences between the best assignment orders are more definite. Considering results for ACO_{heur}, the assignment orders are ranked in the same order, except for DCS and DPD which switch places.

The quality of the best solutions produced by $\text{ACO}_{\text{undir}}$ and ACO_{heur} under different assignment orders is strongly related to the proportion of feasible solutions produced. This result is to be expected given that the quality of solutions produced by an undirected search (or a search biased slightly towards components likely to produce a feasible solution) is dependent on the number of solutions that search encounters.

When pheromone is used a different relationship emerges between the proportion of feasible solutions produced and the quality of those solutions. Table 7.17 shows the results of Mann-Whitney tests comparing assignment orders in terms of minimum RPD and the proportion of feasible solutions produced when heuristic information is not used. Examination of the relative performance of assignment orders shows that amongst the heuristic assignment orders a high proportion of feasible solutions is associated with a lower solution value. When heuristic information is not used the assignment orders may be ranked in terms of the proportion of feasible solutions produced (from least to most) as

$$\text{SFO} \prec \text{DRO} \preceq \text{DPD} \prec \text{DCS} \preceq \text{DPS} \prec \text{SMC} \prec \text{DMC}$$

and in terms of the quality of the best solution produced (worst to best) as

$$\text{SFO} \preceq \text{SMC} \preceq \text{DMC} \prec \text{DCS} \prec \text{DRO} \preceq \text{DPD} \preceq \text{DPS}$$

where $A \prec B$ indicates the difference is statistically significant while $A \preceq B$ indicates that it is not. When heuristic information is used, the order in terms of feasible solutions produced changes little, becoming

$$\text{DRO} \preceq \text{DPD} \preceq \text{DPS} \prec \text{SFO} \prec \text{DCS} \prec \text{SMC} \preceq \text{DMC}$$

with the most notable change being the much improved performance of SFO, which is equivalent to using no assignment order heuristic. However, the cost of solutions produced using SFO remains the worst, as this ranking in terms of quality of the best solution produced shows:

$$\text{SFO} \prec \text{DMC} \prec \text{SMC} \prec \text{DCS} \prec \text{DPD} \preceq \text{DRO} \preceq \text{DPS}$$

Table 7.17: Pairwise comparisons of assignment orders for the GAP when using $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone when heuristic information is not used. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

	minimum RPD					
	DRO	SMCp	DMCa	DCS	DPSp	DPDp
SFO	> (α =1%)	>	>	> (α =1%)	> (α =1%)	> (α =1%)
DRO		< (α =1%)	< (α =1%)	< (α =1%)	>	>
SMCp			>	> (α =1%)	> (α =1%)	> (α =1%)
DMCa				> (α =1%)	> (α =1%)	> (α =1%)
DCS					> (α =1%)	> (α =1%)
DPSp						<

	feasible solutions					
	DRO	SMCp	DMCa	DCS	DPSp	DPDp
SFO	< (α =5%)	< (α =1%)	< (α =1%)	< (α =1%)	< (α =5%)	<
DRO		< (α =1%)	< (α =1%)	< (α =1%)	< (α =5%)	<
SMCp			<	> (α =1%)	> (α =1%)	> (α =1%)
DMCa				> (α =1%)	> (α =1%)	> (α =1%)
DCS					<	> (α =10%)
DPSp						> (α =10%)

It should be noted that when heuristic information was not used, under the majority of assignment orders no algorithm was able to find feasible solutions to the `gap8-1` instance. The exceptions were DRO and DPD, which were able to find two and one feasible solution(s) respectively in total across all 10 random seeds. Hence their ability to find any feasible solutions to this instance appears to be a chance result. When heuristic information was used, all assignment orders were able to produce feasible solutions to this instance.

The relative performance of different assignment orders in terms of solution quality clearly varies little between using and not using heuristic information. However, the quality of solutions produced by ACO under every assignment order was better when heuristic information was used. This result is likely due to the larger number of feasible solutions produced when heuristic information is used, as it favours "safer" assignments as opposed to more profitable ones. Furthermore, the approximately inverse relationship between the proportion of feasible solutions produced and solution quality suggests that the best solutions to these instances are found near the bounds of feasible space. Hence, there can be a trade-off between seeking high quality solutions and keeping the probability of producing feasible solutions high. A similar finding was made by Randall (2004) who, in addition to using a dynamic randomised assignment order, used an adaptive heuristic measure that probabilistically favours assignments with either low cost (the GAP instances studied are minimisation instances) or low resource utilisation.

7.2.6 CSeqP

The CSeqP instances studied are summarised in Table 7.18, and are those studied by Smith et al. (1996).

Pheromone Representation Comparative Performance

The four pheromone representations used with the CSeqP were $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$, $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$, $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$ (same model), where $\mathfrak{C}_{it}$ is the set of production line sequence positions and $\mathfrak{C}_{res}$ is the set of car models. The first pheromone is typical of those used for assignment problems, while the second is a representation-oriented pheromone that may be used because solutions are sequences of elements from $\mathfrak{C}_{res}$. For the second pheromone the elements of $\mathfrak{C}_{res}$ are augmented by an element corresponding to the empty solution so that it can learn which model to assign to the first sequence position. The last two are second order representations where the learned utility of assigning a particular model to a sequence position is based on what other sequence positions have been assigned the same model. Given the length of their descriptive names, for the remainder of this thesis the abbreviations $\mathsf{PH}_{\text{assign-pairs}}$ and $\mathsf{PH}_{\text{same-model}}$ are used for $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$ (same model) respectively. Using the framework introduced in Section 4.2

Table 7.18: CSeqP instances.

Instance	Cars	Models	Optimal cost
n20t1	20	4	58
n20t2	20	4	40
n20t3	20	4	29
n20t4	20	4	10
n20t5	20	4	150
n40t1	40	4	146
n40t2	40	4	94
n40t3	40	4	66
n40t4	40	4	33
n40t5	40	4	352
n60t1	60	4	238
n60t2	60	4	152
n60t3	60	4	105
n60t4	60	4	58
n60t5	60	4	562
n80t1	80	4	330
n80t2	80	4	215
n80t3	80	4	146
n80t4	80	4	82
n80t5	80	4	772

they have the following details. Given a set of solution characteristics $C = \mathfrak{C}_{it} \times \mathfrak{C}_{res}$, the pheromone associated with adding the assignment $(i, r) \in C$ to the partial solution $\mathfrak{s}^p$ is

$$\tau\big(\mathfrak{s}^p, (i,r)\big) = \begin{cases} mean\big(\mathfrak{s}^p, (i,r), \tau_2\big) & \text{if } |C_{(i,r)}| > 0 \\ \tau_1(i,r) & \text{otherwise} \end{cases} \tag{7.5}$$

where τ_2 is the pheromone representation describing pairs of solution characteristics (i.e., either $\mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$ or $\mathfrak{C}^2_{it}$ (same model)) and τ_1 is the first order $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone representation. The function *mean* is defined differently for the two pheromones because the first maintains, for each model, the learned utility of assigning pairs of sequence positions that model, while the second maintains, for each pair of sequence positions, a single value representing the learned utility of assigning them the same model. Hence, for $\mathsf{PH}_{\text{assign-pairs}}$, *mean* is defined as

$$mean\big(\mathfrak{s}^p, (i,r), \tau_2\big) = \frac{\sum_{i' \in C_{(i,r)}} \tau_2(i, i', r)}{|C_{(i,r)}|} \tag{7.6}$$

where $\tau_2(i, i', r)$ is the learned utility of assigning a car of model r to both sequence positions i and i', while for $\mathsf{PH}_{\text{same-model}}$, $mean$ is defined as

$$mean\big(\mathfrak{s}^p, (i, r), \tau_2\big) = \frac{\sum_{i' \in C_{(i,r)}} \tau_2(i, i')}{|C_{(i,r)}|} \tag{7.7}$$

where $\tau_2(i, i')$ is the learned utility of assigning sequence positions i and i' the same model. In both definitions, $C_{(i,r)} = \{i' \in \mathfrak{C}_{it} \mid (i', r) \in s_{\mathfrak{s}^p}\}$. That is, all sequence positions in the partial solution assigned the same model as the candidate. The mean of relevant pheromone values was used because the number of relevant values can differ between models (and hence taking the sum would be inappropriate) and both learn similar information to that used with the GCP, for which the ACO algorithm of Costa and Hertz (1997) uses the mean pheromone value.

It should be noted that $\mathsf{PH}_{\text{same-model}}$ is the pheromone described in Section 6.2.4 as an example of shared solution representations in pheromone.

To assess the sensitivity of the algorithm to the assignment order used, two assignment orders were used:

Static, fixed order (SFO) A static assignment order where sequence positions are assigned in order, from first to last.

Dynamic, randomised order (DRO) The next sequence position to assign is chosen randomly according to a uniform distribution.

As $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ requires a static assignment order to be sensibly applied it is only involved in comparisons within results for SFO.

Table 7.19 shows the results of Mann-Whitney tests comparing alternative approaches when using $\mathcal{MM}$AS, the better performing of the two ACO algorithms. Although comparisons for ACS are not shown, it should be noted that when using ACO with SFO, the performance of $\mathsf{PH}_{\text{assign-pairs}}$ was worse than *all* other alternatives. Examination of the distributions of solution costs for this pheromone for both ACS and $\mathcal{MM}$AS reveals that the algorithm is unable to converge when using SFO. The performance of all approaches was improved when using DRO. Using this assignment order, the relative performances of alternative pheromones under ACS and $\mathcal{MM}$AS are similar, with the exception that the difference between $\mathsf{PH}_{\text{assign-pairs}}$ and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ is not statistically significantly different under ACS. Additionally, using ACS with DRO, $\mathsf{PH}_{\text{assign-pairs}}$ outperforms $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ on large instances (60–80 cars), although the reverse is true under $\mathcal{MM}$AS.

When using SFO with the best performing ACO algorithm, $\mathcal{MM}$AS, the alternatives may be ranked in terms of solution cost as

$$\mathfrak{C}_{it} \times \mathfrak{C}_{res} \prec \mathsf{PH}_{\text{same-model}} \prec \mathsf{PH}_{\text{assign-pairs}} \prec \mathfrak{C}_{res} \times \mathfrak{C}_{res} \prec \text{ACO}_{\text{undir}}$$

Table 7.19: Pairwise comparisons of pheromone representations for the CSeqP when using $\mathcal{MM}$AS. The direction of any difference is shown with the significance level below if the result is statistically significant based on a Mann-Whitney test.

Using SFO

	$\mathsf{PH}_{\text{assign-pairs}}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathsf{PH}_{\text{same-model}}$	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$
$\text{ACO}_{\text{undir}}$	$>$	$>$ (α =1%)	$>$	$>$
$\mathsf{PH}_{\text{assign-pairs}}$		$>$ (α =1%)	$=$	$<$
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$			$<$ (α =1%)	$<$ (α =1%)
$\mathsf{PH}_{\text{same-model}}$				$<$

Using DRO

	$\mathsf{PH}_{\text{assign-pairs}}$	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	$\mathsf{PH}_{\text{same-model}}$
$\text{ACO}_{\text{undir}}$	$>$ (α =1%)	$>$ (α =1%)	$>$
$\mathsf{PH}_{\text{assign-pairs}}$		$>$ (α =1%)	$<$ (α =1%)
$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$			$<$ (α =1%)

while when using DRO with $\mathcal{MM}$AS they may be ranked as

$$\mathfrak{C}_{it} \times \mathfrak{C}_{res} \quad \prec \quad \mathsf{PH}_{\text{assign-pairs}} \quad \prec \quad \mathsf{PH}_{\text{same-model}} \quad \prec \quad \text{ACO}_{\text{undir}}$$

It should be noted that when using DRO, both $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ or $\mathsf{PH}_{\text{assign-pairs}}$ often achieved minimum solution costs within 1–5% of the optimum, even on large instances, while the alternatives were significantly worse, with costs often greater than 20% more than the optimum. On a small number of instances $\mathsf{PH}_{\text{assign-pairs}}$ found better solutions than $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$.

Notably, the best performing pheromone for this problem was $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and not the second order pheromone suggested in Chapter 6, although both performed well. Although both have the property of unique representation, in the previous chapter $\mathsf{PH}_{\text{assign-pairs}}$ was suggested as the most appropriate as it represents how the cost of a particular model at a particular location in the production sequence is dependent on the locations of other cars of the same model. In contrast, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ learns the utility of placing a particular model at a particular position in the production sequence regardless of the location of other cars of the same model. However, if ACO is considered to be searching for a single high quality solution, then modelling which car model belongs at each position in the sequence does largely model characteristics that determine solution cost. Examination of the distribution of solution costs produced by $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ reveals that, on average, almost 40% of solutions produced have the same cost and it is therefore likely that a large proportion of these are the same solution. Consequently, it does appear that $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ converges onto a single good solution for each instance to which it is applied.

7.3 Summary

This chapter presented the results of comparisons between alternative pheromone representations for a number of well-known optimisation problems to test the hypothesis that the best performing pheromone representations are those that model solution characteristics that are directly associated with solution cost or value. Additionally, the performances of a number of assignment orders for the GAP were compared.

The TSP and QAP are both without constructed solution biases, so the alternative pheromone representations used with each are free from any inherent bias. However, the two alternatives for each problem do not perform equivalently. With both problems, the best performing pheromone is that which models solution characteristics that directly contribute to solution cost, namely $\mathfrak{C} \times \mathfrak{C}$ for the TSP and $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ for the QAP. With the TSP, this result holds regardless of whether heuristic information or local search is used, although the performance of all alternatives is improved by the use of both heuristic information and local search. When local search is used with the QAP, $\text{ACO}_{\text{undir}}$, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ and $\mathfrak{C}_{res} \times \mathfrak{C}_{res}$ were all able to find optimal or near optimal solutions for every problem to

which they were applied. However, $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ performed slightly better on two of the three largest instances with which local search was used and converges more quickly to its best solution. This suggests that multiple, shorter runs of $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ with local search should outperform $\text{ACO}_{\text{undir}}$ with local search.

As JSP and GSP instances have a construction bias, any alternative pheromone representations for those problems will exhibit a bias. Furthermore, JSP, GSP and OSP instances typically have a representation bias. As shown in Section 5.2, pheromone representations for the JSP and GSP show predictable patterns of bias, with $\mathfrak{C} \times \mathfrak{C}$ showing a distinct bias towards poor solutions and $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) showing a clear bias towards good solutions. Consequently, it is not surprising that the former was found in this study to perform worse than $\text{ACO}_{\text{undir}}$ while the latter performed best on these problems. As the OSP has no construction bias all three pheromones are CBSs when applied to this problem. Certainly, $\mathfrak{C} \times \mathfrak{C}$ shows no bias towards poor solutions on this problem. However, the order of alternatives in terms of performance does not change, with $\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$ (related succeeding) still outperforming $\mathfrak{C} \times P$, which outperforms $\mathfrak{C} \times \mathfrak{C}$ and $\text{ACO}_{\text{undir}}$.

The MKP also has both a representation and construction bias, so pheromone representations for it are not CBSs, although no predictable biased behaviour could be identified in the analysis of this problem presented in Section 5.3.1. Results for this problem are interesting as, although the suggested pheromone representation $\mathfrak{C}$ performs consistently well, the best results were obtained by a pheromone previously unused with the MKP, $\mathfrak{C} \times P$. It should be noted that, unlike the GSP, where the solution characteristics modelled by representation-oriented pheromones are loosely related to the solution represented, $\mathfrak{C} \times \mathfrak{C}$ and $\mathfrak{C} \times P$ pheromones applied to the MKP do implicitly model which items are included in the solution. Considering $\mathfrak{C} \times P$, it is clear that pheromone that would typically be associated with a single characteristic in $\mathfrak{C}$ is split across values associated with each position that an item may occupy in a solution. It is unclear how this might help the algorithm to find better solutions than if a single pheromone value were associated with each item. The algorithm's sensitivity to the actual ACO algorithm used and whether or not heuristic information is used need to be explored further. It is worth noting that the second order version of $\mathfrak{C}$ also performed well, although often not as well as $\mathfrak{C}$.

The GAP, like most assignment problems, has no representation bias. However, agent capacity constraints introduce a construction bias, which on all the instances studied also includes infeasible partial solutions. Analyses in Section 5.3.2 indicate that solution characteristics associated with assignments that make relatively little use of agent capacity occur in more solutions, yet because no clear relationship exists between capacity utilisation and assignment value there is accordingly no clear relationship between solution characteristic frequency and solution value. Nevertheless, the best performing pheromone for this

problem is that associated with assignments, rather than the order in which resources are assigned, a structural feature of solutions not directly related to the assignments made. This pheromone produced the best solutions regardless of the assignment order or whether or not heuristic information was used.

The assignment order used when solving the GAP clearly has an impact on the number of feasible solutions produced and the quality of solutions found. Assigning tasks in the same order as they appear in an instance description is consistently worst on both measures. When choosing an alternative scheme for determining the assignment order, a balance must be sought between seeking an adequate number of feasible solutions and seeking high quality solutions. Typically, with the exception of SFO, those assignment orders associated with the best quality solutions are also associated with the smallest number of feasible solutions. However, this result may be related to the way the ACO algorithm explores the search space: if the best solutions are found near the bounds of feasible space then an algorithm exploring that region will likely encounter more infeasible solutions. This interaction between the assignment order and reinforcement behaviour of ACO requires further exploration.

The CSeqP solved here was in effect modelled as a GAP in which each model's "capacity" is the number of cars of that model, while each sequence position requires one unit of its assigned model's "capacity". Accordingly, it has no representation bias, but does have a construction bias due to models each having fewer cars than sequence positions. Unlike the GAP it has no infeasible partial solutions. Across the two assignment orders used, the first order $\mathfrak{C}_{it} \times \mathfrak{C}_{res}$ pheromone outperforms the alternatives. The suggested second order pheromone performs very poorly when sequence positions are assigned in strict order, but improves significantly when the order of assignment is randomised. When using a dynamic randomised assignment order, both the first and second order pheromones that are associated with assignments perform well (achieving results close to optimal), and considerably better than representation-oriented alternatives and $\text{ACO}_{\text{undir}}$. Notably, the suggested pheromone, which most accurately captures those solution characteristics that determine cost, is not as effective as its first order counterpart. Further investigation is therefore required into alternative pheromone representations for problems whose objective functions are suggestive of more complex higher order pheromone representations to determine if the use of higher order information is warranted when a first order pheromone representation is available.

In summary, results of the comparative studies of alternative pheromone representations for the TSP, GSP, QAP, GAP and CSeqP all indicate that identity-oriented pheromone representations produce the best performance in ACO. With the TSP, GSP, QAP and GAP, these identity-oriented pheromones also model solution characteristics directly related to

solution cost or value. However, with the CSeqP, the best performing identity-oriented pheromone does not model solution characteristics that individually contribute to solution cost, suggesting that although the identity-oriented second order pheromone does model such characteristics, it does not offer the best means of learning which assignments to make. More generally, this suggests that the use of second order pheromone representations when a first order alternative exists may be unnecessary. Results for the MKP suggest that although the identity-oriented pheromone that models solution characteristics associated with solution value can perform well, it may be subject to premature convergence. Further investigation of pheromone representations for this problem is required to understand why a representation-oriented pheromone is able to outperform an identity-oriented pheromone.

Chapter 8

Conclusions and Further Work

8.1 Summary

ACO is a maturing field of algorithms for combinatorial optimisation. Since its inception in the early 1990s it has been applied to a wide range of combinatorial problems, often with considerable success. A key aspect of its adaptation to different problems is the pheromone representation used to model the solution characteristics added to partial solutions at each step of the algorithm. Much of the ACO literature considers the development of a construction graph for the problem to be solved as essential to the application of ACO as it is a graph based shortest path problem that real ants solve when travelling from the nest to a food source. However, as the algorithm has been applied to an increasing range of problems that have little similarity with shortest path problems, novel pheromone representations have had to be developed which model characteristics unrelated to the construction graph used. Typically, these pheromone representations have been developed in an *ad hoc* way, either by selecting a representation that intuitively appears to model important solution characteristics or that has been used with similar problems previously. Although some research has attempted to understand and predict ACO's behaviour for a given problem and pheromone representation, relatively little of this has examined how it should be adapted to suit different problems. The *ad hoc* adaptation of ACO to some problems has produced algorithms that do not perform to their potential. Consequently, there is a need to adapt ACO using a more rigorous approach that takes into account factors that will influence the algorithm's performance.

Despite the important role of pheromone representations in the adaptation of ACO, there has been no consistent notation for describing them. The notation developed in this thesis provides a consistent and simple way to describe pheromone representations in terms of the solution characteristics they model and is well-suited to the majority of existing ACO applications. It is of particular benefit when systematically selecting pheromone

representations based on a problem model.

The space of solutions explored by constructive metaheuristics such as ACO may be viewed in a number of ways. A particularly informative view is the construction tree, representing both the space of feasible sequences of solution components and the relationship between individual decisions made during the solution construction process. Examination of the construction trees for certain problems reveals that the choice of solution components (and the resulting mapping from sequences of components to solutions) and problem constraints can combine to introduce a solution bias. While alternative problem models may alter these biases, many are a direct result of problem constraints and so cannot be eliminated. Nevertheless, being aware of these biases may enable the development of improved constructive search algorithms, including ACO.

In problems that have unavoidable infeasible solutions, such as most GAP instances, these infeasible solutions have a probability of being produced that is above what would be expected given their number. In assignment problems of this type, altering the order in which items are assigned alters the number and distribution of infeasible paths in the construction tree. A commonly used assignment order heuristic is to assign tightly constrained items early, which intuitively offers them a wider choice of assignments. Notably, such an approach actually increases the probability of individual infeasible paths in the construction tree by decreasing the number of steps before they are reached. However, it also produces considerably fewer such infeasible paths and thereby reduces the probability of producing an infeasible solution. Analysis of a number of assignment order heuristics for the GAP reveals that both static and dynamic heuristics that assign highly constrained items early can achieve a high probability of producing feasible solutions compared to what is possible given problem constraints.

As instance size grows, the effects of low level constructed solution biases become relatively small overall (given the large number of solutions that may be constructed), even though the relative differences between solutions becomes larger. However, in an ACO algorithm, constructed solution biases interact with the pheromone representation used. Critically, if a problem has a construction bias, then there does not exist a pheromone representation that is a competition-balanced system (CBS), in that individual solution characteristics do not all appear with the same frequency in the construction tree. Furthermore, some combinations of pheromone representation and problem that may be CBSs are not free from bias, as a representation bias can still favour some solutions.

Some problems, notably the JSP, have an interesting structure that interacts with different pheromone representations in predictable ways. While the poor performance of $\mathsf{PH}_{\mathsf{suc}}$ and good performance of $\mathsf{PH}_{\mathsf{rel}}$ with this problem has been previously established empirically, and in relation to CBSs, a complete explanation of the underlying causes

requires consideration of both the topology of the construction tree and the way solution characteristics from $\mathsf{PH}_{\mathsf{suc}}$ map onto the tree's edges. Neither $\mathsf{PH}_{\mathsf{suc}}$ nor $\mathsf{PH}_{\mathsf{rel}}$ is a CBS when applied to the JSP, so both exhibit a bias. However, $\mathsf{PH}_{\mathsf{suc}}$ has a relatively large proportion of solution characteristics that are associated with decisions that, in addition to often being part of poor solutions, make other decisions of a similar kind more likely. In contrast, the distribution of solution characteristics from $\mathsf{PH}_{\mathsf{rel}}$ is such that characteristics of poor solutions are often updated only by those solutions, while poor solutions also include characteristics that exist in good solutions.

Other problems, such as the MKP and GAP, also show predictable patterns of solution characteristic usage frequency, but solution characteristics are apparently not as strongly associated with solutions of either high or low value, and so no clear bias can be predicted.

Although altering the way in which pheromone values are updated can be used to reduce bias, the choice of which pheromone representation to use cannot be ignored. To this end, it has been proposed that pheromone representations be chosen such that the solution characteristics they model are directly related to solution cost or value. Many representation-oriented pheromones represent the same solution using differing sets of solution characteristics—a feature that in itself may be undesirable—and indirectly model those characteristics that most directly determine solution cost or value.

An algorithm for the selection of pheromone representations has been proposed that ensures that the solution characteristics modelled are directly related to solution cost or value and consequently that each solution is modelled by exactly one set of solution characteristics. The pheromone representations it selects are therefore identity-oriented. While such an approach cannot eliminate bias, as an identity-oriented pheromone applied to a problem with either a representation or construction bias cannot be a CBS, modelling characteristics that are strongly linked to solution cost or value will in many cases allow the algorithm to learn more effectively and potentially counteract any bias. In the absence of knowledge of how alternative pheromone representations will react to an underlying bias, it therefore offers a practical alternative with identifiable advantages. If the effects of bias are known, this knowledge may be used to override the default selection.

Comparisons of alternative pheromones for the TSP, MKP, GSP, QAP, GAP and CSeqP reveals that, in most cases, identity-oriented pheromone representations outperform representation-oriented alternatives. In the TSP, GSP and GAP, these results hold even if heuristic information or, in the case of the TSP, local search is used. Interestingly, results for the MKP show that when heuristic information is not used, or when the ACO algorithm is a $\mathcal{MM}$AS, a $\mathfrak{C} \times P$ pheromone representation can outperform the intuitively more appropriate $\mathfrak{C}$ pheromone. Indeed, the $\mathfrak{C} \times P$ pheromone representation has never been proposed for use with the MKP. However, when using ACS with heuristic informa-

tion, both $\mathfrak{C}$ and its second order counterpart outperform $\mathfrak{C} \times P$. Results for the QAP reveal that when local search is not used, the suggested pheromone can outperform an alternative pheromone as well as an undirected search. When local search is used, all three alternatives perform equivalently on smaller instances, finding the optimum in every case, although the suggested pheromone performs better on two of the three largest instances with which local search was used. When combining either pheromone representation for the QAP with local search, very fast convergence to one solution was observed, especially when using the pheromone associated with assignments. This suggests that when local search is used a number of short runs of ACO may be a more effective way of sampling the search space and may lead to better performance than an equivalent number of local searches starting from randomly constructed solutions.

With the CSeqP, the two identity-oriented pheromones outperform two representation-oriented alternatives when a dynamic randomised assignment order is used. The suggested pheromone, a second order pheromone describing pairs of sequence positions assigned the same particular model, performs very badly when positions are assigned in their original order, while the first order pheromone associated with assignments still performs well. Notably, when a randomised assignment order is used, the second order pheromone in most cases performs slightly worse than its first order counterpart, suggesting that the higher level information it provides may not be of benefit to the algorithm. Thus, even though the first order pheromone does not learn exactly those solution characteristics that contribute to cost (as an assignment on its own makes no direct contribution to cost) it still appears to offer the most effective way to learn which assignments to make to produce a good solution.

8.2 Contributions

This thesis has made a number of contributions to the field, which are summarised below:

- Solution biases that affect constructive algorithms have been identified. The *representation bias*, which may affect iterative search algorithms as well as constructive, is typically overlooked in the ACO literature. The *construction bias* is peculiar to constructive algorithms and has also not been discussed previously. The underlying causes of both biases have been identified and the biases present in the MKP, GSP and GAP have been discussed. Knowledge of these two biases also allows for their presence to be identified in problems not discussed in the thesis and hence for the more informed application of ACO to other problems.

- A more thorough understanding of the effects of different assignment orders when solving the GAP has been developed, showing that assignment order heuristics that

assign tightly constrained items early increase the proportion of feasible solutions that can be produced by consolidating infeasible paths nearer to the root of the construction tree. Analysis of a number of relatively simple static and dynamic assignment order heuristics has shown they can lead to the production of a high number of feasible solutions.

- These biases, being a product of problem constraints and the construction process, will also be present in other constructive heuristics and metaheuristics, such as GRASP. The findings are therefore not limited to ACO alone.

- A notation for describing pheromone representations in terms of the solution characteristics they model has been developed which can be applied to the majority of ACO algorithms in the literature. A simple framework for describing higher order pheromone representations has been described which makes clearer the steps required in their implementation. Both of these make the direct comparison of different pheromone representations easier. The notation also supports the algorithm for selecting pheromone representations developed in the thesis.

- The relationship between constructed solution biases and the previously defined concept of a CBS has been established. This allows ACO algorithm developers to make more informed choices regarding the pheromone representations they use with particular problems.

- A complete explanation for the relative performances of $\mathsf{PH}_{\mathsf{suc}}$ and $\mathsf{PH}_{\mathsf{rel}}$ applied to the GSP has been provided, complementing previous research in this area. An initial analysis of pheromones for the MKP and GAP suggests that they do not have an obvious bias. The techniques used in all three investigations may prove useful in the analysis of other problems and pheromone representations.

- The distinction between representation- and identity-oriented pheromone representations has been defined and the potential benefits of identity-oriented pheromones presented. In particular, such pheromones represent each solution by exactly one set of solution characteristics and so each solution has a single learned value. Furthermore, the solution characteristics modelled will be directly related to solution cost, so the algorithm should learn more directly than if a representation-oriented pheromone were used.

- An algorithm has been developed for the selection of pheromone representations based on characteristics of the problem being solved. The algorithm uses the objective function to identify which solution characteristics are directly related to solution

cost. It has been applied to a number of COPs, including some to which ACO is yet to be applied.

8.3 Conclusions and Future Work

The ACO metaheuristic can produce good quality solutions to COPs if it is applied correctly. In particular, the pheromone representation used must be chosen carefully. The possible presence of constructed solution biases in a problem must be considered by ACO algorithm developers when choosing how solutions are built and which pheromone representation to use. The structure inherent in the JSP (and non-OSP GSP instances) and its interaction with particular pheromone representations suggests that other problems may have a similarly exploitable structure (or a structure that cannot be exploited and which promotes poor solutions). Initial analysis of the MKP and GAP suggests that pheromone representations for these problems do not have an obvious bias, but new approaches to investigating the potential for bias need to be explored and other problems must also be studied.

As part of the continued exploration of bias in ACO algorithms, the role of local search must be examined in more detail. Part of this involves the study of available local search techniques and the expected bias they exhibit, as different local search procedures explore the search space in different ways and will consequently give varying access to different regions. Additionally, there is often a trade-off between the efficiency and effectiveness of a local search (Dorigo and Stützle, 2004). While it is important to study the best way to apply ACO and local search individually, the interaction between the two cannot be ignored. For instance, it is conceivable that the use of a powerful local search procedure to improve solutions before they are used to update pheromone information may lead to rapid, and perhaps premature, convergence of the algorithm. Consequently research in this area must consider how to achieve the best balance between exploitation of the best solution characteristics found using local search and exploration of a range of different, possibly less high quality, solutions.

In parallel with the further study of bias in ACO, extensions may be made to the notation for describing pheromone representations and the framework for higher order pheromones. Currently these are concerned primarily with static features of pheromone representations and thus could be extended to encompass the dynamic behaviour of pheromone and how it is used and updated. The higher order pheromone framework takes some initial steps in this direction, but more detail is needed. In particular, further investigation is required to identify how best to use higher order information given the problem being solved, an issue which also involves the study of bias.

The decision algorithm presented in Chapter 6 represents an initial step towards the systematic selection of pheromones based on the problem being solved. As the pheromone representation notation is extended to include the dynamic use of pheromone information this can be included as part of the decision algorithm's suggestions. Given that the algorithm is intended to identify solution characteristics that are strongly associated with determining solution cost, its recommendations will likely correspond to those of a human algorithm developer. However, there are some problems for which it would have difficulty identifying a single most appropriate pheromone representation. One such class of problems are multiple objective problems, an example being the SMTTP with changeover costs (Iredi et al., 2001). In order to handle these problems, separate pheromone representations can be used for each objective (this was done by Iredi et al.) and the algorithm can then be applied to each objective separately. Another class of problems for which a single pheromone representation may not be appropriate are multiple task problems, such as the p-hub median class of problems (Ernst and Krishnamoorthy, 1996). These problems model distribution networks and consist of two stages of solution construction, one to identify nodes to become hubs, and a second to identify which hub each other node is to use when routing commodities through the network. The two stages are thus separate, but interacting. Again it may be possible to use separate pheromone representations for each stage. However, as these problems have a single objective, the algorithm would require modification to allow it to suggest pheromone representations for different parts of a single objective.

Future versions of the algorithm, in addition to supporting a wider range of problems, can be implemented in software and will take in objective functions written in a suitable modelling language. This modelling language will likely be quite separate from the sequences ants produce so that important relationships between different parts of solutions can be more easily identified. Given an implementation of the algorithm, the pheromone selection software could be incorporated into a self-adapting ACO implementation, thereby allowing for the rapid adaptation and application of ACO to new problems.

Bibliography

Anderson, D. R., Sweeney, D. J. and Williams, T. A. (1994). *An introduction to management science: Quantitative approaches to decision making*, 7th edn, West Publishing Co., Minneapolis.

Bauer, A., Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999). Applying ant colony optimization to solve the single machine total tardiness problem, *Technical Report Nr. 42*, Vienna University of Economics and Business Administration, Vienna, Austria.

Beasley, J. E. (2005). OR-Library, http://people.brunel.ac.uk/˜mastjjb/jeb/info.html.

Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M. and Abramson, D. (2000). Scheduling aircraft landings - the static case, *Transportation Science* **34**(2): 180–197.

Berry, L. T. M., Murtagh, B. A., McMahon, G. B., Sugden, S. and Welling, L. (1999). An integrated GA-LP approach to communication network design, *Telecommunications Systems* **12**(2): 265–280.

Bertsekas, D. P. (1999). *Nonlinear Programming*, 2nd edn, Athena Scientific, Belmont, MA.

Birattari, M., Di Caro, G. and Dorigo, M. (2002). Toward the formal foundation of ant programming, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 188–201.

Blum, C. (2002a). ACO applied to group shop scheduling: A case study on intensification and diversification, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 14–27.

Blum, C. (2002b). Ant colony optimization for the edge-weighted k-cardinality tree problem, *in* W. Langdon (ed.), *Genetic and Evolutionary Computation Conference (GECCO 2002)*, New York, USA, Morgan Kaufmann Publishers, pp. 27–34.

Blum, C. (2004). *Theoretical and practical aspects of ant colony optimization*, Vol. 282 of *Dissertations in Artificial Intelligence*, IOS Press, Nieuwe Hemweg, The Netherlands.

Blum, C. and Dorigo, M. (2003). The hyper-cube framework for ant colony optimization, *IEEE Transactions on Systems, Man and Cybernetics–Part B* **34**(2): 1161–1172.

Blum, C., Roli, A. and Dorigo, M. (2001). HC-ACO: The hyper-cube framework for ant colony optimization, *4th Metaheuristics International Conference, MIC'2001*, Porto, Portugal, pp. 399–403.

Blum, C. and Sampels, M. (2002a). Ant colony optimization for FOP shop scheduling: A case study on different pheromone representations, *2002 Congress on Evolutionary Computation*, pp. 1558–1563.

Blum, C. and Sampels, M. (2002b). When model bias is stronger than selection pressure, *in* J. J. Merelo-Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañas and H.-P. Schwefel (eds), *7th International Conference on Parallel Problem Solving from Nature (PPSN2002)*, Vol. 2439 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 893–902.

Blum, C. and Sampels, M. (2004). An ant colony optimization algorithm for shop scheduling problems, *Journal of Mathematical Modelling and Algorithms* **3**(3): 285–308.

Bonabeau, E., Dorigo, M. and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1997). Applying the ant system to the vehicle routing problem, *2nd Metaheuristic International Conference, MIC97*, Sophia-Antipolis, France.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999a). An improved ant system algorithm for the vehicle routing problem, *Annals of Operations Research* **89**: 319–328.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999b). A new rank based version of the ant system: A computational study, *Central European Journal for Operations Research and Economics* **7**(1).

Burkard, R. E., Çela, E., Karisch, S. E. and Rendl, F. (2004). QAPLIB - a quadratic assignment problem library, http://www.seas.upenn.edu/qaplib/.

Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P. and Schulenburg, S. (2002). Hyper-heuristics: An emerging direction in modern search technology, *in* F. Glover and G. Kochenberger (eds), *Handbook of Meta-Heuristics*, Vol. 57 of *International*

Series in Operations Research and Management Science, Kluwer Academic Publishers, Boston, MA, pp. 457–474.

Chu, P. C. and Beasley, J. E. (1998a). Constraint handling in genetic algorithms: The set partitioning problem, *Journal of Heuristics* **4**(4): 323–357.

Chu, P. C. and Beasley, J. E. (1998b). A genetic algorithm for the multidimensional knapsack problem, *Journal of Heuristics* **4**: 63–86.

Cohen, J. and Stewart, I. (1994). *The Collapse of Chaos: Discovering Simplicity in a Complex World*, Penguin, London.

Colorni, A., Dorigo, M., Maniezzo, V. and Trubian, M. (1994). Ant system for job-shop scheduling, *JORBEL - Belgian Journal of Operations Research, Statistics and Computer Science* **34**(1): 39–53.

Cordón, O., Fernández de Viana, I. and Herrera, F. (2002a). Analysis of the best-worst ant system and its variants on the QAP, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 228–234.

Cordón, O., Fernández de Viana, I. and Herrera, F. (2002b). Analysis of the best-worst ant system and its variants on the TSP, *Mathware and Soft Computing* **9**(2): 177–192.

Cordón, O., Fernández de Viana, I., Herrera, F. and Moreno, L. (2000). A new ACO model integrating evolutionary computation concepts: The best-worst ant system, *2nd International Workshop on Ant Algorithms (ANTS'2000)*, Brussels, Belgium, pp. 22–29.

Corne, D., Dorigo, M. and Glover, F. (eds) (1999). *New Ideas in Optimization*, Advanced Topics in Computer Science, McGraw-Hill, London.

Costa, D. and Hertz, A. (1997). Ants can colour graphs, *Journal of the Operational Research Society* **48**: 295–305.

Darwin, C. (1859, reprinted in 1988). *On the Origin of Species*, New York University Press, New York.

Dawkins, R. (1986). *The Blind Watchmaker*, Penguin, London.

den Besten, M., Stützle, T. and Dorigo, M. (2000). Ant colony optimization for the total weighted tardiness problem, *in* M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo and H. Schwefel (eds), *6th International Conference on Parallel*

Problem Solving from Nature (PPSN-VI), Paris, France, Vol. 1917 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 611–620.

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

Doerner, K., Gronalt, M., Hartl, R. F., Reimann, M., Strauss, C. and Stummer, M. (2002). Savingsants for the vehicle routing problem, *in* S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf and G. R. Raidl (eds), *EvoWorkshops 2002*, Kinsale, Ireland, Vol. 2279 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 11–20.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms (in Italian)*, Phd thesis, Politecnico di Milano.

Dorigo, M., Bonabeau, E. and Theraulaz, G. (2000). Ant algorithms and stigmergy, *Future Generation Computer Systems* **16**: 851–871.

Dorigo, M. and Di Caro, G. (1999). The ant colony optimization meta-heuristic, *in* D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, pp. 11–32.

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

Dorigo, M. and Gambardella, L. M. (1997a). Ant colonies for the traveling salesman problem, *BioSystems* **43**: 73–81.

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

Dorigo, M., Maniezzo, V. and Colorni, A. (1991). Ant system: An autocatalytic optimizing process, *Technical Report 91-016 Revised*, Politecnico di Milano, Milan, Italy.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man and Cybernetics–Part B* **26**(1): 1–13.

Dorigo, M. and Stützle, T. (2002). The ant colony optimisation metaheuristic: Algorithms, applications and advances, *in* F. Glover and G. Kochenberger (eds), *Handbook of Metaheuristics*, Vol. 57 of *International Series in Operations Research and Management Science*, Kluwer Academic Publishers, Boston, MA, pp. 251–285.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*, MIT Press.

Ducatelle, F. and Levine, J. (2001). Ant colony optimisation for bin packing and cutting stock problems, *UK Workshop on Computational Intelligence (UKCI-01)*, Edinburgh.

Ducatelle, F. and Levine, J. (2004). Ant colony optimisation and local search for bin packing and cutting stock problems, *Journal of the Operational Research Society* **55**(7): 705–716.

Ellabib, I., Otman, A. B. and Calamai, P. (2002). An experimental study of a simple ant colony system for the vehicle routing problem with time windows, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 53–64.

Ernst, A. T. and Krishnamoorthy, M. (1996). Efficient algorithms for the uncapacitated single allocation p-hub median problem, *Location Science* **4**(3): 139–154.

Fenet, S. and Solnon, C. (2003). Searching for maximum cliques with ant colony optimization, *in* G. Raidl, S. Cagnoni, J. Romero Cardalda, D. Corne, J. Gottlieb, A. Guillot, E. Hart, C. Johnson, E. Marchiori, J.-A. Meyer and M. Middendorf (eds), *EvoWorkshops 2003*, Essex, UK, Vol. 2611 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 236–245.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**: 109–133.

Fiorenzo Catalamo, M. S. and Malucelli, F. (2001). Parallel randomized heuristics for the set covering problem, *in* M. Paprzycki, L. Tarricone and L. T. Yang (eds), *Practical Parallel Computing*, Nova Science Publishers, Inc., pp. 113–132.

Fogel, D. B. (1995). Phenotypes, genotypes, and operators in evolutionary computation, *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation*, Perth, Australia, IEEE Press, pp. 193–198.

Fogel, L. J. (1997). A retrospective view and outlook on evolutionary algorithms, *in* B. Reusch (ed.), *Computational Intelligence: Theory and Applications, 5th Fuzzy Days*, Springer-Verlag, Berlin, pp. 337–342.

Fogel, L. J. (1999). *Intelligence Through Simulated Evolution: Forty Years of Evolutionary Programming*, Wiley Series on Intelligent Systems, John Wiley, New York.

Forsyth, P. and Wren, A. (1997). An ant system for bus driver scheduling, *7th International Workshop on Computer-Aided Scheduling of Public Transport*, Boston, USA, pp. 405–421.

Gagné, C., Price, W. L. and Gravel, M. (2002). Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times, *Journal of the Operational Research Society* **53**(8): 895–906.

Gambardella, L. M. and Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem, *in* A. Prieditis and S. Russell (eds), *12th International Conference on Machine Learning (ML-95)*, Tahoe City, CA, Morgan Kaufmann, pp. 252–260.

Gambardella, L. M. and Dorigo, M. (1996). Solving symmetric and asymmetric TSPs by ant colonies, *IEEE Conference on Evolutionary Computation (ICEC96)*, Nagoya, Japan, pp. 622–627.

Gambardella, L. M. and Dorigo, M. (1997). HAS-SOP: An hybrid ant system for the sequential ordering problem, *Technical Report IDSIA-11-97*, IDSIA, Lugano, Switzerland.

Gambardella, L. M. and Dorigo, M. (2000). An ant colony system hybridized with a new local search for the sequential ordering problem, *Journal on Computing* **12**(3): 237–255.

Gambardella, L. M., Taillard, É. D. and Agazzi, G. (1999). MACS-VRPTW - a multiple ant colony system for vehicle routing problems with time windows, *in* D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, pp. 63–76.

Gambardella, L. M., Taillard, É. D. and Dorigo, M. (1999). Ant colonies for the quadratic assignment problem, *Journal of the Operational Research Society* **50**: 167–176.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York.

Glover, F. and Kochenberger, G. (eds) (2002). *Handbook of Metaheuristics*, Vol. 57 of *International Series in Operations Research and Management Science*, Kluwer Academic Publishers, Boston, MA.

Glover, F. and Laguna, M. (1997). *Tabu Search*, Kluwer Academic Publishers, Boston, MA.

Gottlieb, J., Puchta, M. and Solnon, C. (2003). A study of greedy, local search, and ant colony optimization approaches to car sequencing problems, *in* G. Raidl, S. Cagnoni, J. Romero Cardalda, D. Corne, J. Gottlieb, A. Guillot, E. Hart, C. Johnson, E. Marchiori, J.-A. Meyer and M. Middendorf (eds), *EvoWorkshops 2003*, Essex, UK, Vol. 2611 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 246–257.

Grassé, P.-P. (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la théorie de la stigmergie: essai dinterprétation du comportement des termites constructeurs, *Insectes Sociaux* **6**: 41–81.

Guntsch, M. and Middendorf, M. (2002a). Applying population based ACO to dynamic optimization problems, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 111–122.

Guntsch, M. and Middendorf, M. (2002b). A population based approach for ACO, *in* S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf and G. R. Raidl (eds), *EvoWorkshops 2002*, Kinsale, Ireland, Vol. 2279 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 72–81.

Gutjahr, W. J. (2000). A graph-based ant system and its convergence, *Future Generation Computer Systems* **16**: 873–888.

Gutjahr, W. J. (2002). ACO algorithms with guaranteed convergence to the optimal solution, *Information Processing Letters* **82**(3): 145–153.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, Ann Arbor.

Iredi, S., Merkle, D. and Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms, *1st International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, Zurich, Vol. 1993 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 359–372.

Jaffar, J. and Maher, M. J. M. (1994). Constraint logic programming: A survey, *Journal of Logic Programming* **19–20**(Supplement 1): 503–581.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation; part ii, graph coloring and number partitioning, *Operations Research* **39**(3): 378–405.

Johnson, D. S. and McGeoch, L. A. (1997). The traveling salesman problem: A case study, *in* E. Aarts and J. Lenstra (eds), *Local Search in Combinatorial Optimization*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, Chichester, UK.

Kennedy, J. and Eberhart, R. C. (1995). Particle swarm optimization, *IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942–1948.

Lawrence, S. (1984). Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (supplement), *Technical report*, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.

Leguizamón, G. and Michalewicz, Z. (1999). A new version of ant system for subset problems, *1999 Congress on Evolutionary Computation*, pp. 1459–1464.

Lourenço, H. R. and Serra, D. (2002). Adaptive search heuristics for the generalized assignment problem, *Mathware and Soft Computing* **9**(2): 209–234.

Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS Journal on Computing* **11**(4): 358–369.

Maniezzo, V. and Carbonaro, A. (2000). An ants heuristic for the frequency assignment problem, *Future Generation Computer Systems* **16**: 927–935.

Maniezzo, V. and Colorni, A. (1999). The ant system applied to the quadratic assignment problem, *IEEE Transactions on Knowledge and Data Engineering* **11**(5): 769–778.

Maniezzo, V. and Milandri, M. (2002). An ant-based framework for very strongly constrained problems, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 222–227.

Merkle, D. and Middendorf, M. (2000). An ant algorithm with a new pheromone evaluation rule for total tardiness problems, *in* S. Cagnoni, R. Poli, G. Smith, D. Corne, M. Oates, E. Hart, P. Lanzi, E. Willem, Y. Li, B. Paechter and T. Fogarty (eds), *EvoWorkshops 2000*, Vol. 1803 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 287–296.

Merkle, D. and Middendorf, M. (2001). A new approach to solve permutation scheduling problems with ant colony optimisation, *in* E. J. W. Boers, S. Cagnoni, J. Gottlieb, E. Hart, P. L. Lanzi, G. Raidl, R. E. Smith and H. Tijink (eds), *Applications of Evolutionary Computing, EvoWorkshops 2001*, Como, Italy, Vol. 2037 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 484–494.

Merkle, D. and Middendorf, M. (2002). Ant colony optimization with the relative pheromone evaluation method, *in* S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf and G. Raidl (eds), *EvoWorkshops 2002*, Kinsale, Ireland, Vol. 2279 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 321–329.

Merkle, D., Middendorf, M. and Schmeck, H. (2000). Ant colony optimization for resource-constrained project scheduling, *in* D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee and H.-G. Beyer (eds), *2000 Genetic and Evolutionary Computation Conference*, Las Vegas, NV, USA, Morgan Kaufmann Publishers, pp. 893–900.

Meuleau, N. and Dorigo, M. (2002). Ant colony optimization and stochastic gradient descent, *Artificial Life* **8**(2): 103–121.

Michalewicz, Z. (1996). *Genetic algorithms + Data structures = Evolution Programs*, 3rd edn, Springer - Verlag, New York.

Michel, R. and Middendorf, M. (1999). An ant system for the shortest common supersequence problem., *in* D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, pp. 51–61.

Monmarché, N., Ramat, E., Dromel, G., Slimane, M. and Venturini, G. (1999). On the similarities between as, BSC and PBIL: Toward the birth of a new meta-heuristic, *Technical Report 215*, Ecole d'Ingénieurs en Informatique pour l'Industrie (E3i), Université de Tours.

Montgomery, J. and Randall, M. (2002). Anti-pheromone as a tool for better exploration of search space, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 100–110.

Montgomery, J., Randall, M. and Hendtlass, T. (2004). Search bias in constructive meta-heuristics and implications for ant colony optimisation, *in* M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada and T. Stützle (eds), *4th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2004*, Brussels, Belgium, Vol. 3172 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 390–397.

Nowé, A. and Verbeeck, K. (1999). Formalizing the ant algorithms in terms of reinforcement learning, *in* D. Floreano, J.-D. Nicoud and F. Mondada (eds), *Advances in Artificial Life, 5th European Conference, ECAL'99*, Lausanne, Switzerland, Vol. 1674 of *Lecture Notes in Computer Science*, Springer, pp. 616–620.

Osman, I. H. and Kelly, J. P. (eds) (1996). *Meta-heuristics: Theory & Applications*, Kluwer Academic Publishers, Boston.

Petersen, C. C. (1967). Computational experience with variants of the balas algorithm applied to the selection of R&D projects, *Management Science* **13**(9): 736–750.

Picton, P. (1994). *Introduction to Neural Networks*, Macmillan, Basingstoke.

Rahoual, M., Hadji, R. and Bachelet, V. (2002). Parallel ant system for the set covering problem, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 262–267.

Randall, M. (1999). *A General Modelling System and Meta-heuristic based Solver for Combinatorial Optimisation Problems*, Phd thesis, Griffith University.

Randall, M. (2000). Using simulated annealing to solve telecommunication network design problems, *APORS 2000: The 5th Conference of the Association of Asian-Pacific Operational Research Societies Within IFORS*, Singapore.

Randall, M. (2002a). A general framework for constructive meta-heuristics, *in* E. Kozan and A. Ohuchi (eds), *Operations Research/Management Science at Work*, Vol. 43 of *International Series in Operations Research and Management Science*, Kluwer Academic Publishers, Boston, MA, pp. 111–128.

Randall, M. (2002b). Scheduling aircraft landings using ant colony optimisation, *6th IASTED International Conference Artificial Intelligence and Soft Computing*, Banff, Alberta, Canada, pp. 129–133.

Randall, M. (2003a). A systematic strategy to incorporate intensification and diversification into ant colony optimisation, *in* H. A. Abbass and J. Wiles (eds), *1st Australian Conference on Artificial Life (ACAL03)*, Canberra, Australia, University of New South Wales, pp. 199–208.

Randall, M. (2003b). A template approach to producing incremental cost functions for local search meta-heuristics, *in* G. Raidl, S. Cagnoni, J. Romero Cardalda, D. Corne, J. Gottlieb, A. Guillot, E. Hart, C. Johnson, E. Marchiori, J.-A. Meyer and M. Middendorf (eds), *EvoWorkshops 2003*, Essex, UK, Vol. 2611 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 291–299.

Randall, M. (2004). Heuristics for ant colony optimisation using the generalised assignment problem, *Proceedings of the Congress on Evolutionary Computing 2004*, Portland, OR, USA.

Randall, M. and Abramson, D. (2001). A general meta-heuristic based solver for combinatorial optimisation problems, *Journal of Computational Optimization and Applications* **20**(2): 185–210.

Randall, M. and Montgomery, J. (2002). Candidate set strategies for ant colony optimisation, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 243–249.

Randall, M. and Tonkes, E. (2001). Solving network synthesis problems using ant colony optimisation, *in* L. Monostori, J. Vancza and M. Ali (eds), *14th International Conference on Industrial and Engineering Applications of Artificial Intelligence*, Vol. 2070 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 1–10.

Randall, M. and Tonkes, E. (2002). Intensification and diversification strategies in ant colony system, *Complexity International* **9**: online at http://journal–ci.csse.monash.edu.au.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, Stuttgart.

Reeves, C. R. (1999). Landscapes, operators and heuristic search, *Annals of Operations Research* **86**: 473–490.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*, Vol. 840 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin.

Reinelt, G. (2004). TSPlib - a traveling salesman problem library, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPlib95/.

Resende, M. G. C., Pitsoulis, L. S. and Pardalos, P. (1997). Approximate solution of weighted max-sat problems using GRASP, *in* D.-Z. Du, J. Gu and P. Pardalos (eds), *Satisfiability Problem: Theory and Applications*, Vol. 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, American Mathematical Society, pp. 393–405.

Roli, A., Blum, C. and Dorigo, M. (2001). ACO for maximal constraint satisfaction problems, *4th Metaheuristics International Conference*, Porto, Portugal, pp. 187–192.

Schoofs, L. and Naudts, B. (2000). Solving CSPs with ant colonies, *Abstract Proceedings of ANTS2000*, Brussels, Belgium.

Schwefel, H.-P. (1975). *Evolutionsstrategie und Numerische Optimierung*, Dissertation, Technical University of Berlin.

Shmygelska, A., Aguirre-Hernández, R. and Hoos, H. (2002). An ant colony optimization algorithm for the 2d protein folding problem, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 40–52.

Smith, K., Palaniswami, M. and Krishnamoorthy, M. (1996). A hybrid neural network approach to combinatorial optimisation, *Computers and Operations Research* **73**: 501–508.

Socha, K., Knowles, J. and Sampels, M. (2002). A $\mathcal{MAX} - \mathcal{MIN}$ ant system for the university course timetabling problem, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS 2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 1–13.

Solnon, C. (2000). Solving permutation constraint satisfaction problems with ant algorithms, *in* W. Horn (ed.), *14th European Conference on Artificial Intelligence*, Berlin, Germany, IOS Press, pp. 118–122.

Solnon, C. (2002). Ants can solve constraint satisfaction problems, *IEEE Transactions on Evolutionary Computation* **6**(4).

Stewart, I. and Cohen, J. (1994). Why are there simple rules in a complex universe?, *Futures* **26**: 648–664.

Stützle, T. (1997). $\mathcal{MAX} - \mathcal{MIN}$ ant system for the quadratic assignment problem, *Technical Report AIDA-97-04*, Darmstadt University of Technology, Computer Science Department, Intellectics Group.

Stützle, T. (1998). An ant approach to the flow shop problem, *6th European Congress on Intelligent Techniques & Soft Computing (EUFIT '98)*, Aachen, Germany, Verlag Mainz, pp. 1560–1564.

Stützle, T. and Dorigo, M. (1999a). ACO algorithms for the quadratic assignment problem., *in* D. Corne, M. Dorigo and F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, pp. 33–50.

Stützle, T. and Dorigo, M. (1999b). ACO algorithms for the traveling salesman problem, *in* K. Miettinen, M. Makela, P. Neittaanmaki and J. Periaux (eds), *Evolutionary Algorithms in Engineering and Computer Science*, Wiley, pp. 163–183.

Stützle, T. and Dorigo, M. (2002). A short convergence proof for a class of ant colony optimization algorithms, *IEEE Transactions on Evolutionary Computation* **6**(4): 358–365.

Stützle, T. and Hoos, H. (1996). Improving the ant system: A detailed report on the $\mathcal{MAX} - \mathcal{MIN}$ ant system, *Technical Report AIDA-96-12 - Revised version*, Darmstadt University of Technology, Computer Science Department, Intellectics Group.

Stützle, T. and Hoos, H. (1998). The $\mathcal{MAX} - \mathcal{MIN}$ ant system and local search for combinatorial optimization problems: Towards adaptive tools for combinatorial global optimization, *in* S. Voss, S. Martello, I. Osman and C. Roucairol (eds), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Boston, MA, pp. 313–329.

Stützle, T. and Hoos, H. (2000). $\mathcal{MAX} - \mathcal{MIN}$ ant system, *Future Generation Computer Systems* **16**: 889–914.

Stützle, T. and Linke, S. (2002). Experiments with variants of ant algorithms, *Mathware and Soft Computing* **9**(2): 193–207.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA.

Taha, H. (1992). *Operations Research: An Introduction*, 5th edn, Macmillan Publishing Company, New York.

Taillard, É. D. and Gambardella, L. M. (1997). Adaptive memories for the quadratic assignment problem, *Technical Report IDSIA-87-97*, IDSIA, Lugano, Switzerland.

T'kindt, V., Monmarché, N., Tercinet, F. and Laügt, D. (2002). An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem, *European Journal of Operational Research (EJOR)* **142**(2): 250–257.

van der Zwaan, S. and Marques, C. (1999). Ant colony optimisation for job shop scheduling, *3rd Workshop on Genetic Algorithms and Artificial Life (GAAL 99)*.

van Laarhoven, P. J. M. and Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications*, D. Reidel Publishing Company, Dordrecht, Holland.

Vander Meer, R. K., Breed, M. D., Winston, M. L. and Espelie, K. E. (eds) (1997). *Pheromone Communication in Social Insects: Ants, Wasps, Bees, and Termites*, Westview Press, Boulder, Colorado.

Watkins, C. J. (1989). *Learning from Delayed Rewards*, Phd thesis, University of Cambridge.

Watkins, C. J. and Dayan, P. (1992). Q-learning, *Machine Learning* **8**(3): 279–292.

Winston, P. (1992). *Artificial Intelligence*, 3rd edn, Addison Wesley, Reading, MA.

Winston, W. (1991). *Operations Research: Applications and Algorithms*, 2nd edn, Duxbury Press, Belmont, CA.

Zlochin, M. and Dorigo, M. (2002). Model-based search for combinatorial optimization: A comparative study, *7th International Conference on Parallel Problem Solving from Nature (PPSN2002)*, pp. 651–662.

Appendix A

Glossary of Problem Names and Acronyms

This glossary contains brief descriptions of optimisation problems and their associated acronyms for those problems that appear more than once in the thesis. With the exception of the term *a priori*, italicised terms in descriptions appear elsewhere in the glossary.

2DHPPF. See *2D HP Protein Folding*.

2D HP Protein Folding. An *assignment problem* in which the shape an amino acid chain will fold into is predicted using the two-dimensional hydrophobic-polar model of protein folding.

AIRLAND. See *Aircraft Landing Problem*.

Aircraft Landing Problem. An *assignment problem* in which aircraft must be assigned landing times on one or more runways such that minimum separation constraints between different aircraft are respected. The aim is to minimise the difference between each aircraft's actual and most economical landing times.

Assignment Problem. Any problem where a set of items must be assigned one or more resources, subject to certain constraints.

Asymmetric Travelling Salesman Problem. Equivalent to a symmetric *TSP* with the exception that distances between a pair of nodes may differ depending on the direction of travel.

ATSP. See *Asymmetric Travelling Salesman Problem*.

Bin Packing Problem. An *assignment problem* in which a number of items of different weights must be packed into a number of bins all with the same capacity. The aim is to minimise the number of bins required.

BPP. See *Bin Packing Problem.*

BUS. See *Bus Driver Scheduling.*

Bus Driver Scheduling. A problem in which drivers must be assigned to periods of work such that all work is covered. The aim is to minimise the number of drivers required.

Car Sequencing Problem. An *assignment problem* in which a number of different car models must be placed in a production sequence. The aim is separate cars of the same model by the greatest distance possible.

Constraint Satisfaction Problem. An *assignment problem* in which a number of variables must be assigned values such that a number of logical constraints involving those variables are satisfied, or that the number of unsatisfied constraints is minimised.

CSatP. See *Constraint Satisfaction Problem.*

CSeqP. See *Car Sequencing Problem.*

CStockP. See *Cutting Stock Problem.*

Cutting Stock Problem. An *assignment problem* in which a number of items must be cut from stocks of the same length. The aim is to minimise the number of stocks required.

FAP. See *Frequency Assignment Problem.*

Flow Shop Problem. A *scheduling problem* in which a number of jobs must be processed on each of a number of machines in the same order. The aim is to determine the order in which jobs are given access to each machine such that the makespan is minimised.

Frequency Assignment Problem. An *assignment problem* in which a number of requests for frequencies in a radio network must be serviced. The aim is to minimise the amount of interference between radio links.

FSP. See *Flow Shop Problem.*

GAP. See *Generalised Assignment Problem.*

GCP. See *Graph Colouring Problem.*

Generalised Assignment Problem. An archetypal *assignment problem* in which each of a number of tasks must be assigned to one of a number of agents such that each agent's capacity can support the tasks assigned. The aim is either to minimise the cost or maximise the profit of the assignments made.

GPP. See *Graph Partitioning Problem.*

Graph Colouring Problem. An *assignment problem* in which the nodes of a graph must be assigned colours such that no two adjacent nodes have the same colour. The problem may be posed with the aim of either minimising the number of colours required to produce a feasible colouring or minimising the number of like-coloured adjacent nodes for a given number of colours.

Graph Partitioning Problem. An *assignment problem* in which a graph must be partitioned into two sets of equal size. The aim is to minimise the number of edges between nodes in different partitions.

GSP. See *Group-shop Scheduling Problem.*

Group-shop Scheduling Problem. A generalisation of the *JSP* and *OSP* in which each job's operations are partitioned into groups such that pre-existing precedence constraints exist between groups, but not between operations within each group. The aim is to minimise the makespan.

Job-shop Scheduling Problem. A *scheduling problem* in which a number operations, each of which must be processed on exactly one of a number of machines, must be scheduled for processing. Operations are partitioned into jobs and the processing order of operations within each job is specified *a priori*. The aim is to minimise the makespan.

JSP. See *Job-shop Scheduling Problem.*

KCTP. See *k-Cardinality Tree Problem.*

***k*-Cardinality Tree Problem.** A *subset problem* where a tree consisting of k edges must be formed by taking a subset of edges from an edge- or node-weighted graph. The aim is to minimise the weight of included edges or nodes.

Linear Ordering Problem. A problem in which a number of items must be ordered such that the cost due to certain items preceding certain other items is minimised.

LOP. See *Linear Ordering Problem.*

Maximum Clique Problem. A *subset problem* in which a clique (i.e., a set of fully connected nodes) is sought in a graph. The aim is to maximise the size of the clique.

MCP. See *Maximum Clique Problem.*

MKP. See *Multiple Knapsack Problem.*

Multiple Knapsack Problem. A *subset problem* in which a subset of items is sought from some larger set such that their weights do not exceed the capacities of a number of knapsacks. Items often represent projects and knapsacks available resources. The aim is to maximise the profit from the included items/projects.

NETSYNTH. See *Network Synthesis Problem.*

Network Synthesis Problem. A problem in which a network topology must be devised, followed by the allocation of bandwidth between pairs of nodes in the network, such that the required bandwidth between nodes is available. The aim is to minimise both the cost of constructing links between nodes and the ongoing costs of using the allocated bandwidth on those links.

NPP. See *Number Partitioning Problem.*

NQP. See *N-Queens Problem.*

***N*-Queens Problem.** An *assignment problem* in which N queens must be placed on an N-by-N chess board such that no two queens can attack each other.

Number Partitioning Problem. An *assignment problem* in which a set of numbers must be split into two partitions of equal size. The aim is to minimise the difference between the sums of numbers in each partition.

OSP. See *Open-shop Scheduling Problem.*

Open-shop Scheduling Problem. A *scheduling problem* in which a number operations, each of which must be processed on exactly one of a number of machines, must be scheduled for processing. Operations are partitioned into jobs, but the order of operations within each job is not specified *a priori*. The aim is to minimise the makespan.

PAP. See Processor Allocation Problem.

Processor Allocation Problem. An *assignment problem* in which a number of processes must be assigned to processors in a multi-processing computing environment. The aim is to minimise the amount of inter-processor communication.

QAP. See *Quadratic Assignment Problem.*

Quadratic Assignment Problem. An *assignment problem* in which each of n facilities must be assigned to exactly one of n locations such that the product of the distance and flow between facilities is minimised.

Scheduling Problem. A combinatorial optimisation problem where operations must be scheduled for processing on machines. Typically the aim is to minimise the makespan, or time to complete all jobs.

SCP. See *Set Covering Problem.*

SCSP. See *Shortest Common Supersequence Problem.*

Sequential Ordering Problem. A problem in which a Hamiltonian path of minimum weight is sought in a node- and edge-weighted graph.

Set Covering Problem. A *subset problem* in which a subset of columns, each of which covers some of a number of constraints, must be chosen from some larger set such that all constraints are covered. The aim is to minimise the cost of the columns used.

Set Partitioning Problem. A *subset problem* in which a subset of columns, each of which covers some of a number of constraints, must be chosen from some larger set such that each constraint is covered by exactly one column. The aim is to minimise the cost of the columns used.

Shortest Common Supersequence Problem. A problem in which a string that is a supersequence of a number of other strings (i.e., any of those other strings may be obtained by deleting characters from the solution string) is sought. The aim is to minimise the length of the supersequence.

Single Machine Total Tardiness Problem. A *scheduling problem* in which a number of operations, all requiring the same machine, must be placed in a sequence. The aim is to minimise the total amount by which operations are late.

SMTTP. See *Single Machine Total Tardiness Problem.*

SOP. See *Sequential Ordering Problem.*

SPP. See *Set Partitioning Problem.*

Subset Problem. Any problem where a subset of items from some larger set is sought, subject to certain constraints.

Symmetric Travelling Salesman Problem. A problem in which a Hamiltonian circuit of minimum weight is sought in an undirected edge-weighted graph.

TSP. See *Symmetric Travelling Salesman Problem.*

UCTP. See *University Course Timetabling Problem.*

University Course Timetabling Problem. An *assignment problem* in which events such as classes must be assigned to timeslots. The problem considered in this thesis contains a number of soft constraints that relate to good timetable design, with the aim of minimising the number of soft constraint violations.

Vehicle Routing Problem. A problem in which a number of customers must serviced by a number of vehicles with fixed capacities such that either the total number of vehicles required or the total distance travelled/time taken to service all customisers is minimised.

VRP. See *Vehicle Routing Problem*.

Appendix B

Objective Functions Used in Pheromone Representation Selection

The algorithm for automatically selecting pheromone representations presented in Chapter 6 was applied to the following problem models. Note that problem constraints have been omitted as the algorithm does not consider these.

Bin Packing Problem The following objective function minimises the amount of constraint violation, rather than the number of bins used.

$$Minimise \sum_{i=1}^{M} \max \left(0,\ \sum_{j=1}^{|B(i)|} w\big(A(i,j)\big) - W_{max} \right) \tag{B.1}$$

Where:

$B(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of items assigned to bin i, where k is an integer and $I(k)$ is the i^{th} item to be assigned during construction.

$A(i, j) \in B(i)$ is the j^{th} item assigned to bin i.

W_{max} is the maximum weight each bin may hold.

$w(i)$ is the weight of item i.

M is the number of bins.

N is the number of items.

Car Sequencing Problem Based on the car sequencing problem described by Smith et al. (1996).

$$Minimise \sum_{i=1}^{M} \sum_{j=1}^{|D(i)|-1} \sum_{k=j+1}^{|D(i)|} P\Big(|A(i,k) - A(i,j)|, i\Big) \tag{B.2}$$

Where:

$D(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of sequence positions assigned to model i, where k is an integer and $I(k)$ is the i^{th} sequence position to be assigned during construction.

$A(i, j) \in D(i)$ is the j^{th} sequence position assigned to model i.

$P(i, j)$ is the separation penalty for the j^{th} model separated by i places in the sequence.

N is the number of cars.

M is the number of models.

Cutting Stock Problem The following objective function minimises the amount of constraint violation, rather than the number of stocks used.

$$Minimise \sum_{i=1}^{M} \sum_{j=1}^{|S(i)|} r\big(A(i,j)\big) - R_{max} \tag{B.3}$$

Where:

$S(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of pieces to be cut from stock i, where k is an integer and $I(k)$ is the i^{th} piece to be assigned during construction.

$A(i, j) \in S(i)$ is the j^{th} piece assigned to stock i.

R_{max} is the length of each stock.

$r(i)$ is the length of stock required by piece i.

N is the number of pieces to be produced.

M is the number of stocks available.

Frequency Allocation Problem The Frequency Allocation Problem involves the assignment of frequencies to requests (i.e., links in a radio network). The following objective function minimises the degree of interference resulting from the frequencies assigned.

$$Minimise \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \max\Big(0,\ reqddist\big(I(i), I(j)\big) - \big|\mathfrak{s}[i] - \mathfrak{s}[j]\big|\Big) \tag{B.4}$$

Where:

$I(i) \in \mathfrak{C}_{it}$ is the i^{th} request assigned a frequency.

$\mathfrak{s}[i] \in \mathfrak{C}_{res}$ is the frequency assigned to request $I(i)$.

N is the number of frequency requests.

M is the number of frequencies available.

$reqddist(i, j)$ is the minimum distance required between frequencies assigned to requests i and j, below which interference will occur.

$\max(i, j)$ returns i if $i > j$, j otherwise.

Generalised Assignment Problem The following formulation of the GAP is as a maximisation problem, which corresponds to the instances studied. If the instance is a minimisation problem then the objective changes to minimise and the interpretation of $C(i, j)$ is the cost of assigning item i to agent j.

$$Maximise \sum_{i=1}^{N} C\big(I(i), \mathfrak{s}[i]\big) \tag{B.5}$$

Where:

$C(i, j)$ is the profit of assigning item i to agent j.

$I(i) \in \mathfrak{C}_{it}$ is the i^{th} task assigned an agent.

$\mathfrak{s}[i] \in \mathfrak{C}_{res}$ is the agent assigned to task $I(i)$.

N is the number of tasks.

M is the number of agents.

Graph Colouring Problem The following objective function minimises the number of constraint violations, rather than the number of colours used.

$$Minimise \sum_{i=1}^{M} \sum_{j=1}^{|C(i)|-1} \sum_{k=j+1}^{|C(i)|} edge\big(A(i, j), A(i, k)\big) \tag{B.6}$$

Where:

$C(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of nodes assigned colour i, where k is an integer and $I(k)$ is the i^{th} node to be assigned during construction.

$A(i, j) \in C(i)$ is the j^{th} node assigned colour i.

M is the number of colours available.

N is the number of nodes.

$edge(i, j)$ is 1 if there is an edge connecting nodes i and j, 0 otherwise.

Graph Partitioning Problem

$$Minimise \sum_{i=1}^{\frac{N}{2}} \sum_{j=1}^{\frac{N}{2}} edge\big(A(1,i), A(2,j)\big) \tag{B.7}$$

Where:

$A(i, j) \in \mathfrak{C}_{it}$ is the j^{th} node assigned to partition i.

$edge(i, j)$ is 1 if nodes i and j are adjacent, 0 otherwise.

N is the number of nodes.

Group Shop Scheduling Problem The pheromone selection algorithm was applied to the definition of the term $t(i)$.

$$Minimise \max_{i=1,\ldots,N} t(\mathfrak{s}[i]) \tag{B.8}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the operation at the i^{th} position in the solution sequence.

$t(i) = \max_{j \in R(i)} t(j) + p(i)$ is the completion time of operation i.

$R(i) = \{j \in \mathfrak{C} \mid loc(j) < loc(i),\ \mathcal{M}(i) = \mathcal{M}(j) \vee \mathcal{G}(i) = \mathcal{G}(j)\}$ is the set of operations related to and preceding operation i, where $loc(i)$ is the position of operation i in the sequence, $\mathcal{M}(i)$ is the machine required by i and $\mathcal{G}(i)$ is the group to which i belongs.

$p(i)$ is the processing time of i.

$\max_{i=1,\ldots,N} f(i)$ returns the maximum value of the function $f(i)\ \forall\, i = 1, \ldots, N$.

N is the number of operations.

Multiple Knapsack Problem

$$Maximise \sum_{i=1}^{|\mathfrak{s}|} c(\mathfrak{s}[i]) \tag{B.9}$$

Where:

$|\mathfrak{s}|$ is the size of the solution.

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} item included.

$c(i)$ is the profit of including item i.

N is the number of items.

Linear Ordering Problem

$$Minimise \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c(\mathfrak{s}[i], \mathfrak{s}[k]) \tag{B.10}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} item in the sequence.

$c(i, j)$ is the cost of placing item i before item j in the sequence.

N is the number of items.

Maximum Clique Problem This formulation minimises the number of constraint violations for a give clique size, rather than the size of the clique.

$$Minimise \frac{M^2 - M}{2} - \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} edge(\mathfrak{s}[i], \mathfrak{s}[j]) \tag{B.11}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} node in the solution sequence $\mathfrak{s}$.

$edge(i, j)$ is 1 if nodes i and j are adjacent, 0 otherwise.

N is the number of nodes.

M is the clique size.

Number Partitioning Problem

$$Minimise \left| \sum_{i=1}^{|P(1)|} A(1, i) - \sum_{i=1}^{|P(2)|} A(2, i) \right| \tag{B.12}$$

Where:

$P(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1, N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of numbers assigned to partition i, where k is an integer and $I(k)$ is the i^{th} number to be assigned during construction.

$A(i,j) \in P(i)$ is the j^{th} number assigned to partition i.

N is the number of numbers.

N Queens Problem

$$Minimise \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} attacks(\mathfrak{s}[i], \mathfrak{s}[j]) \tag{B.13}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}_{res}$ is the i^{th} board position assigned to queen i.

$attacks(i,j)$ is 1 if queens at positions i and j can attack each other, 0 otherwise.

N is the number of queens.

Processor Allocation Problem

$$Minimise \sum_{i=1}^{M} \sum_{j=1}^{|P(i)|} \sum_{k=i+1}^{M} \sum_{l=1}^{|P(k)|} c\big(A(i,j), A(k,l)\big) \tag{B.14}$$

Where:

$P(i) = \{j \in \mathfrak{C}_{it} \mid k \in [1,N],\ \mathfrak{s}[k] = i,\ j = I(k)\}$ is the set of processes assigned to processor i, where k is an integer and $I(k)$ is the i^{th} process to be assigned during construction.

$A(i,j) \in P(i)$ is the j^{th} process assigned to processor i.

N is the number of processes.

M is the number of processors.

$c(i,j)$ is communication cost between processes i and j.

Quadratic Assignment Problem

$$Minimise \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} f\big(I(i), I(j)\big) \cdot d(\mathfrak{s}[i], \mathfrak{s}[j]) \tag{B.15}$$

Where:

$I(i) \in \mathfrak{C}_{it}$ is the i^{th} facility assigned.

$\mathfrak{s}[i] \in \mathfrak{C}_{res}$ is the location assigned to facility $I(i)$.

$f(i,j)$ is the flow between facilities i and j.

$d(i,j)$ is the distance between locations i and j.

N is the number of facilities/locations.

Set Covering/Partitioning Problems These problems differ only in their constraints, which are not presented here. In both problems the term *column* is typically used instead of the generic term *item* used with regards to subset problems elsewhere in the thesis.

$$Minimise \sum_{i=1}^{|\mathfrak{s}|} c(\mathfrak{s}[i]) \tag{B.16}$$

Where:

$|\mathfrak{s}|$ is the size of the solution.

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} item included.

$c(i)$ is the cost of including column i.

N is the number of columns.

Single Machine Total Tardiness Problem

$$Minimise \sum_{i=1}^{N} \max\left(0,\ \sum_{j=1}^{i} p(\mathfrak{s}[j]) - d(\mathfrak{s}[i])\right) \tag{B.17}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} job to be processed in the sequence.

$p(i)$ is the processing time of job i.

$d(i)$ is the due date of job i.

$\max(i, j)$ returns i if $i > j$, j otherwise.

N is the number of jobs.

Travelling Salesman Problem

$$Minimise \sum_{i=1}^{N} d(\mathfrak{s}[i], \mathfrak{s}[\text{pred}(i, 1, 1, N)]) \tag{B.18}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} city in the solution sequence.

$\text{pred}(i, k, l, u)$ returns the value $i-k$ unless $i-k < l$, in which case u is returned.

N is the number of cities.

Vehicle Routing Problem The following formulation is for the minimisation of the distance travelled only and assumes that a vehicle returning to the depot is modelled as an extra node in the sequence.

$$Minimise \sum_{i=1+1}^{N} d(\mathfrak{s}[i], \mathfrak{s}[i-1]) \tag{B.19}$$

Where:

$\mathfrak{s}[i] \in \mathfrak{C}$ is the i^{th} customer visited in the sequence (or the depot).

N is the number of customers plus the number of artificial depot nodes.

Appendix C

Details of Non-benchmark Problem Instances Used

This appendix provides details of those instances studied that are not available in benchmark problem libraries. The problems are arranged according to the first chapter in which they are extensively studied.

Chapter 3

`jsp2-2`, a two job, two machine, four operation JSP instance. This JSP instance was described by Blum and Sampels (2002b), and has the following specification. The set of operations

$$\mathcal{O} = \{1, 2, 3, 4\},$$

the set of jobs

$$\mathcal{J} = \{J_1 = \{1, 2\}, J_2 = \{3, 4\}\},$$

with $1 \prec 2$, $3 \prec 4$, the set of machines

$$\mathcal{M} = \{M_1 = \{1, 4\}, M_2 = \{2, 3\}\},$$

and processing times of the operations are $p(1) = p(4) = 10$ and $p(2) = p(3) = 20$. $i \prec j$ indicates i must be processed before j. The optimal solution cost is 40.

3 Agent, 8 Task GAP instance. Number of agents $n = 3$, number of tasks $m = 8$. The cost of assigning task i to agent j is C_{ij} where

$$C = \begin{pmatrix} 17 & 21 & 22 & 18 & 24 & 15 & 20 & 18 \\ 23 & 16 & 21 & 16 & 17 & 16 & 19 & 25 \\ 16 & 20 & 16 & 25 & 24 & 16 & 17 & 19 \end{pmatrix}$$

the resource required by task i when assigned to agent j is R_{ij} where

$$R = \begin{pmatrix} 8 & 15 & 14 & 23 & 8 & 16 & 8 & 25 \\ 15 & 7 & 23 & 22 & 11 & 11 & 12 & 10 \\ 21 & 20 & 6 & 22 & 24 & 10 & 24 & 9 \end{pmatrix}$$

and the capacities of agents $b = (35, 35, 35)$.

Chapter 5

`jsp3-3` and `osp3-3` JSP and OSP instances. These instances, based on similar instances described by Blum and Sampels (2002a), have the following specifications. In both instances the set of operations

$$\mathcal{O} = \{1, \ldots, 9\},$$

the set of jobs

$$\mathcal{J} = \{J_1 = \{1, 2, 3\}, J_2 = \{4, 5, 6\}, J_3 = \{7, 8, 9\}\},$$

the set of machines

$$\mathcal{M} = \{M_1 = \{1, 5, 9\}, M_2 = \{2, 6, 7\}, M_3 = \{3, 4, 8\}\},$$

and the processing times of operations is $p(1) = p(5) = p(9) = 10$, $p(2) = p(6) = p(7) = 20$ and $p(3) = p(4) = p(8) = 30$. In `jsp3-3`, $1 \prec 2 \prec 3$, $4 \prec 5 \prec 6$ and $7 \prec 8 \prec 9$, while `osp3-3` has no precedence constraints.

The optimal solution cost for both instances is 90.

Appendix D

Tables of Results

This appendix contains tables of all data summarised and described in Chapter 7, organised by problem type, in the same order presented in that chapter. There are two tables for the TSP, the first for when local search was not used and the second for those cases where it was. These are followed by one table each for the MKP, GSP, QAP, GAP and CSeqP. Each table starts on a new page. Tables are broken across pages such that results for each instance appear on the same page. Table headings indicate if heuristic information or local search were used. The symbol η is used to denote the use of heuristic information, while **LS** is used to denote the use of local search.

Each row reports the minimum (labelled **min**), median (labelled **med**) and maximum (labelled **max**) RPD across all random seeds for a single parameter combination. That is, for each combination of instance, algorithm (labelled **alg.**), pheromone type, heuristic information, local search and assignment order (labelled **assign. order**). Results for the GAP also include the percentage of feasible solutions produced (labelled **% feas.**).

Table D.1: TSP results, no local search.

			no η, no LS			η, no LS		
Instance	Alg.	Pheromone	min	med	max	min	med	max
gr24	ACO_{undir}	—	79.2	178.9	261.4	12.7	83.8	197.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	20.8	243.5	0	0.5	147.9
	ACS	$\mathfrak{C} \times P$	28.9	70.1	241.4	10.6	35.8	158.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	14.8	100.6	235.1	0	3.9	147.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	42.6	155.6	267.9	0	49.3	172.9
hk48	ACO_{undir}	—	188.8	329.0	438.2	47.3	129.5	244.5
	ACS	$\mathfrak{C} \times \mathfrak{C}$	54.0	140.5	422.9	0	7.5	216.7
	ACS	$\mathfrak{C} \times P$	81.9	132.8	424.1	19.9	61.1	216.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	84.7	183.3	407.4	0	0.7	194.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	106.7	237.5	425.6	6.2	56.5	221.2
eil51	ACO_{undir}	—	186.9	288.0	379.3	43.2	132.6	233.1
	ACS	$\mathfrak{C} \times \mathfrak{C}$	56.1	127.7	340.1	1.2	3.5	192.7
	ACS	$\mathfrak{C} \times P$	74.2	123.9	360.1	32.6	64.8	213.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	82.4	167.1	350.0	0.2	2.6	196.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	106.1	219.5	368.5	13.1	61.7	211.0
st70	ACO_{undir}	—	316.7	442.2	555.0	68.9	152.0	268.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	118.5	220.1	521.8	1.3	7.3	209.9
	ACS	$\mathfrak{C} \times P$	139.3	211.0	551.6	49.8	90.8	260.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	153.5	262.4	519.0	0.1	3.0	218.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	182.1	323.0	543.9	24.1	83.6	249.2
eil76	ACO_{undir}	—	267.3	369.1	455.2	79.9	158.4	248.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	104.1	185.5	425.1	2.4	4.8	218.8
	ACS	$\mathfrak{C} \times P$	104.1	179.4	433.6	46.7	82.0	235.5
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	133.6	221.4	437.5	0.6	2.6	218.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	137.2	258.6	446.7	25.3	75.5	234.6
kroA100	ACO_{undir}	—	507.9	704.2	862.0	99.4	192.8	316.2
	ACS	$\mathfrak{C} \times \mathfrak{C}$	240.4	363.9	831.5	3.6	20.2	283.2
	ACS	$\mathfrak{C} \times P$	223.1	345.5	851.4	75.5	122.3	315.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	268.4	423.9	831.5	0.2	3.8	283.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	314.6	491.0	846.6	44.1	122.1	339.3
d198	ACO_{undir}	—	891.0	1109.4	1283.2	90.9	164.3	296.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	415.6	597.1	1248.4	5.1	29.0	235.3
	ACS	$\mathfrak{C} \times P$	374.6	512.4	1248.5	74.3	108.6	267.5
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	455.6	665.1	1249.6	2.9	12.2	236.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	402.2	584.4	1259.5	64.8	120.7	299.9
lin318	ACO_{undir}	—	1139.7	1299.2	1451.8	189.2	293.7	432.0
	ACS	$\mathfrak{C} \times \mathfrak{C}$	756.4	917.0	1443.9	8.5	19.1	370.3
	ACS	$\mathfrak{C} \times P$	626.2	787.3	1451.2	172.8	238.4	464.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	825.1	967.3	1420.8	6.5	9.4	368.3
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	694.2	882.5	1419.9	154.1	245.2	471.7
pcb442	ACO_{undir}	—	40.6	54.3	67.4	59.9	49.9	39.3
	ACS	$\mathfrak{C} \times \mathfrak{C}$	3.7	18.2	64.9	89.1	85.5	42.2
	ACS	$\mathfrak{C} \times P$	15.7	0.8	65.1	66.2	59.9	30.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	8.8	22.8	65.8	89.3	86.0	39.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	5.2	10.7	64.7	66.7	58.0	30.7
att532	ACO_{undir}	—	1574.1	1750.3	1930.2	264.0	358.6	466.2
	ACS	$\mathfrak{C} \times \mathfrak{C}$	1114.3	1290.1	1891.9	11.1	31.6	433.6
	ACS	$\mathfrak{C} \times P$	888.3	1082.7	1889.9	225.1	285.4	554.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	1166.8	1341.6	1888.2	7.6	22.6	423.5
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	905.5	1103.1	1871.3	213.2	304.4	524.6

Table D.2: TSP results, with local search.

			no η, LS			η, LS		
Instance	Alg.	Pheromone	min	med	max	min	med	max
gr24	$\text{ACO}_{\text{undir}}$	—	0	3.7	23.3	0	3.7	25.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	16.6	0	0	18.2
	ACS	$\mathfrak{C} \times P$	0	3.5	24.3	0	2.7	22.3
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0	17.8	0	0	17.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	3.7	30.4	0	3.3	23.7
hk48	$\text{ACO}_{\text{undir}}$	—	0	5.5	18.2	0	5.3	20.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	14.4	0	0	13.6
	ACS	$\mathfrak{C} \times P$	0	5.5	18.7	0	4.6	16.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0	15.3	0	0	16.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	5.5	20.3	0	4.7	17.6
eil51	$\text{ACO}_{\text{undir}}$	—	0	6.1	21.4	0	5.9	20.4
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	14.6	0	0	13.1
	ACS	$\mathfrak{C} \times P$	0	6.1	21.8	0	5.4	18.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0.2	17.6	0	0	13.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	6.1	21.4	0	5.6	20.0
st70	$\text{ACO}_{\text{undir}}$	—	0	5.8	21.0	0	5.5	20.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	17.9	0	0	15.1
	ACS	$\mathfrak{C} \times P$	0	5.6	23.0	0	4.9	20.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0	20.0	0	0	15.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	5.6	23.3	0	5.0	22.4
eil76	$\text{ACO}_{\text{undir}}$	—	0.2	8.4	20.1	0	8.0	20.4
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	17.5	0	0	13.8
	ACS	$\mathfrak{C} \times P$	0.6	8.2	20.6	0	7.4	19.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0	17.5	0	0	17.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.2	8.2	20.8	0.2	7.4	19.7
kroA100	$\text{ACO}_{\text{undir}}$	—	0	6.6	21.9	0	6.3	22.2
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	21.0	0	0	19.8
	ACS	$\mathfrak{C} \times P$	0	6.5	22.7	0	4.9	21.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	0	18.2	0	0	17.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	6.5	22.2	0	5.1	21.5

Table D.3: MKP results.

			no η			η		
Instance	Alg.	Pheromone	min	med	max	min	med	max
mknap1-6item	$\text{ACO}_{\text{undir}}$	—	0	36.8	36.8	0	18.4	36.8
	ACS	$\mathfrak{C}$	0	0	36.8	0	0	36.8
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	2.6	36.8	0	2.6	36.8
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	36.8	0	0	36.8
	ACS	$\mathfrak{C} \times P$	0	0	36.8	0	0	36.8
	$\mathcal{MM}$AS	$\mathfrak{C}$	0	13.2	36.8	0	0	36.8
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	28.9	36.8	0	18.4	36.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	31.6	36.8	0	2.6	36.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	31.6	36.8	0	2.6	36.8
mknap1-10item	$\text{ACO}_{\text{undir}}$	—	0	20.3	47.9	0	0.6	47.9
	ACS	$\mathfrak{C}$	0	0	47.9	0	0	33.5
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	1.5	47.9	0	0	47.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	47.9	0	0	29.2
	ACS	$\mathfrak{C} \times P$	0	0	47.9	0	0	33.5
	$\mathcal{MM}$AS	$\mathfrak{C}$	0	0	47.9	0	0	34.8
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	9.8	47.9	0	0	47.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	9.8	47.9	0	0	47.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	9.8	47.9	0	0	34.8
mknap1-15item	$\text{ACO}_{\text{undir}}$	—	0	21.2	55.2	0	3.1	55.2
	ACS	$\mathfrak{C}$	0	0.2	48.4	0	0	32.0
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	0	53.9	0	0	44.3
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	49.8	0	0	41.6
	ACS	$\mathfrak{C} \times P$	0	0	53.9	0	0	32.9
	$\mathcal{MM}$AS	$\mathfrak{C}$	0	0.2	53.9	0	0	35.9
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	3.1	55.2	0	0	55.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	14.3	55.2	0	0	55.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	13.1	55.2	0	0	44.7
mknap1-20item	$\text{ACO}_{\text{undir}}$	—	0	38.2	64.4	0	7.2	45.7
	ACS	$\mathfrak{C}$	0	0.5	57.0	0	0	32.8
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	3.8	60.4	0	0	40.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0.3	61.2	0	0	30.6
	ACS	$\mathfrak{C} \times P$	0	0.2	61.2	0	0	31.7
	$\mathcal{MM}$AS	$\mathfrak{C}$	0	0.2	60.5	0	0	29.6
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	7.5	64.4	0	0	47.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	12.1	64.4	0	0	39.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	9.3	63.1	0	0	40.9
mknap1-28item	$\text{ACO}_{\text{undir}}$	—	0	25.8	51.3	0.1	7.5	28.2
	ACS	$\mathfrak{C}$	0	1.5	46.1	0.2	0.2	17.9
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	0	46.1	0	0.2	17.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0	0	46.1	0.1	0.2	20.7
	ACS	$\mathfrak{C} \times P$	0	0.1	48.7	0	0.2	17.9
	$\mathcal{MM}$AS	$\mathfrak{C}$	0	0	49.5	0.1	0.2	22.8
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	1.3	47.4	0	0.2	22.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0	15.5	50.9	0	0.2	25.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	9.2	50.0	0	0.2	26.9

continues. . .

Table D.3: MKP results (continued).

			no η			η		
Instance	Alg.	Pheromone	min	med	max	min	med	max
mknap1-39item	ACO_{undir}	—	0.3	22.3	43.1	1.5	14.6	37.4
	ACS	$\mathfrak{C}$	0.5	0.9	37.1	0.8	5.5	22.8
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0.1	1.7	42.4	0	3.7	29.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0.3	1.4	41.6	0	3.5	22.9
	ACS	$\mathfrak{C} \times P$	0.3	1.4	39.5	0	0.6	35.0
	$\mathcal{MM}$AS	$\mathfrak{C}$	0.3	1.7	38.5	0.6	6.1	34.7
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0.3	1.7	41.4	0	3.4	38.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0.3	6.4	42.6	0	6.0	30.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.1	5.7	41.6	0	1.7	32.6
mknap1-50item	ACO_{undir}	—	1.0	22.5	55.5	1.3	14.8	38.2
	ACS	$\mathfrak{C}$	0.5	1.4	48.2	0.7	5.2	23.0
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0.2	0.8	48.2	0.6	1.6	35.7
	ACS	$\mathfrak{C} \times \mathfrak{C}$	0.2	0.8	53.3	0	5.0	33.2
	ACS	$\mathfrak{C} \times P$	0.4	0.7	48.2	0.2	1.6	31.4
	$\mathcal{MM}$AS	$\mathfrak{C}$	0.2	0.8	50.9	1.6	5.2	27.5
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0.2	0.8	52.7	0.8	5.2	32.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0.2	9.9	54.9	0.6	3.9	38.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.1	5.5	53.5	0.3	1.7	33.3
mknapcb1-0.5-1	ACO_{undir}	—	7.6	16.9	33.4	5.6	14.3	30.9
	ACS	$\mathfrak{C}$	2.8	9.0	24.8	0.5	3.7	20.9
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	3.5	11.1	26.7	1.0	6.6	22.0
	ACS	$\mathfrak{C} \times \mathfrak{C}$	1.6	3.7	28.4	1.6	2.1	24.2
	ACS	$\mathfrak{C} \times P$	1.5	3.1	30.3	1.4	2.1	26.1
	$\mathcal{MM}$AS	$\mathfrak{C}$	1.7	7.2	29.7	0.5	1.5	23.4
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	2.6	9.2	28.3	0.9	4.5	23.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	5.2	13.4	30.6	1.9	9.1	26.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.7	7.2	31.5	0.5	3.8	25.5
mknapcb1-0.5-2	ACO_{undir}	—	7.2	17.8	34.3	5.2	14.1	30.6
	ACS	$\mathfrak{C}$	2.9	9.5	27.6	0	2.8	24.1
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	3.7	11.5	27.5	0.6	5.9	24.8
	ACS	$\mathfrak{C} \times \mathfrak{C}$	2.6	3.8	28.2	1.4	1.8	27.7
	ACS	$\mathfrak{C} \times P$	1.7	3.3	29.3	0.6	1.7	25.8
	$\mathcal{MM}$AS	$\mathfrak{C}$	1.8	8.1	28.4	0	1.1	24.1
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	2.1	9.8	27.9	0	4.4	24.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	4.4	13.9	32.6	1.4	8.2	27.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.8	7.0	32.0	0	3.2	31.2
mknapcb1-0.5-3	ACO_{undir}	—	7.1	16.8	33.3	5.8	14.7	31.1
	ACS	$\mathfrak{C}$	2.0	8.5	28.2	0.4	4.1	22.5
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	3.0	10.6	26.7	1.1	6.5	23.0
	ACS	$\mathfrak{C} \times \mathfrak{C}$	2.0	2.7	28.1	1.2	1.8	23.9
	ACS	$\mathfrak{C} \times P$	1.7	2.3	29.2	0.8	2.0	24.0
	$\mathcal{MM}$AS	$\mathfrak{C}$	0.8	6.9	27.3	0.4	3.6	23.0
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	2.2	8.9	27.8	0.6	5.0	26.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	3.8	12.9	31.7	1.9	9.4	28.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	1.0	7.0	31.2	0.4	4.4	27.6

continues. . .

Table D.3: MKP results (continued).

			no η			η		
Instance	Alg.	Pheromone	min	med	max	min	med	max
mknapcb1-0.25-1	$\text{ACO}_{\text{undir}}$	—	10.8	26.1	51.7	8.1	23.3	52.0
	ACS	$\mathfrak{C}$	1.4	10.6	40.4	0	7.0	35.3
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	1.4	13.5	45.8	2.0	11.4	36.3
	ACS	$\mathfrak{C} \times \mathfrak{C}$	2.4	4.3	44.9	2.0	3.6	38.0
	ACS	$\mathfrak{C} \times P$	1.4	3.1	42.1	1.1	2.6	41.3
	$\mathcal{MM}$AS	$\mathfrak{C}$	0.5	7.9	43.9	0.5	5.3	38.2
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	1.5	11.8	41.4	1.5	10.2	41.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	1.1	14.1	47.0	1.2	9.7	41.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	6.1	45.7	0.7	5.5	41.8
mknapcb1-0.25-2	$\text{ACO}_{\text{undir}}$	—	11.4	25.9	47.4	9.5	23.6	46.3
	ACS	$\mathfrak{C}$	1.9	11.7	40.2	0.3	2.3	34.2
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	3.2	14.6	40.4	1.8	11.7	36.1
	ACS	$\mathfrak{C} \times \mathfrak{C}$	2.9	5.3	41.5	1.0	3.0	41.3
	ACS	$\mathfrak{C} \times P$	0.6	2.5	44.0	0.6	3.3	37.8
	$\mathcal{MM}$AS	$\mathfrak{C}$	1.0	9.0	42.9	0.3	1.8	36.4
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	1.5	12.7	41.4	1.1	10.1	40.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	1.4	14.4	44.3	1.6	9.9	42.9
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0	6.4	44.0	0.3	5.4	40.3
mknapcb1-0.25-3	$\text{ACO}_{\text{undir}}$	—	12.6	27.9	51.1	9.9	24.5	48.2
	ACS	$\mathfrak{C}$	1.5	11.4	39.7	0.1	1.9	33.0
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	3.2	14.9	48.9	1.2	11.0	35.8
	ACS	$\mathfrak{C} \times \mathfrak{C}$	3.6	5.8	44.4	0.7	3.3	39.6
	ACS	$\mathfrak{C} \times P$	1.7	3.3	42.4	0.4	2.3	43.6
	$\mathcal{MM}$AS	$\mathfrak{C}$	0.2	7.8	46.0	0.1	1.4	39.5
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	1.3	12.6	46.1	0.3	9.6	43.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	0.6	14.5	44.0	0.2	9.1	41.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	0.2	6.0	45.0	0.1	5.0	43.7

Table D.4: GSP results.

			no η			η		
Instance	Alg.	Pheromone	min	med	max	min	med	max
ft10-1	ACO_{undir}	—	32.8	96.6	193.1	30.8	101.2	202.6
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	9.6	21.5	177.8	12.0	24.7	165.8
	ACS	$\mathfrak{C} \times \mathfrak{C}$	46.9	127.0	299.1	51.0	121.1	299.1
	ACS	$\mathfrak{C} \times P$	31.3	39.1	193.7	21.8	45.7	192.6
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	12.3	22.7	168.7	10.6	26.2	165.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	46.7	207.5	329.6	48.9	194.5	320.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	35.6	98.2	204.0	31.8	100.0	193.4
ft10-5	ACO_{undir}	—	50.5	131.6	262.2	50.8	130.5	276.7
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	25.1	33.4	209.1	14.4	23.5	222.2
	ACS	$\mathfrak{C} \times \mathfrak{C}$	43.3	115.4	406.6	52.1	71.0	376.6
	ACS	$\mathfrak{C} \times P$	28.9	42.6	289.8	27.5	42.8	259.8
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	9.9	25.9	214.6	11.6	17.5	208.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	67.4	184.1	327.1	62.8	180.1	327.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	53.9	138.2	278.1	43.7	122.1	255.9
ft10-10	ACO_{undir}	—	48.5	123.4	252.7	47.3	121.4	255.4
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	21.5	28.2	231.8	10.8	19.1	224.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	44.6	109.5	238.8	30.7	100.5	231.1
	ACS	$\mathfrak{C} \times P$	21.2	28.2	229.3	21.5	28.1	208.7
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	6.3	54.4	233.1	5.5	13.6	213.0
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	42.4	111.1	244.9	36.9	104.6	222.3
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	35.4	97.1	228.4	20.8	82.1	226.0
la38-1	ACO_{undir}	—	54.0	115.9	217.7	53.6	124.4	236.7
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	19.1	24.2	177.5	16.0	27.4	192.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	70.0	528.0	739.8	69.1	509.7	724.8
	ACS	$\mathfrak{C} \times P$	48.0	73.8	281.2	53.3	97.7	272.0
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	13.0	20.2	174.2	10.1	19.1	192.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	58.6	505.4	702.4	72.7	496.5	690.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	62.2	131.1	241.1	63.8	134.3	244.1
la38-8	ACO_{undir}	—	83.5	183.1	323.7	97.2	194.0	340.7
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	40.8	85.3	297.9	26.2	41.7	276.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	117.0	455.9	769.8	112.8	465.1	761.5
	ACS	$\mathfrak{C} \times P$	61.2	83.5	392.5	61.1	79.1	400.4
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	13.8	33.5	297.9	18.0	26.4	271.7
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	101.3	329.6	527.3	116.2	350.4	536.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	108.9	212.3	355.2	96.0	196.5	364.6
la38-15	ACO_{undir}	—	78.2	147.8	268.8	73.3	145.0	269.0
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	53.6	125.2	271.1	17.3	22.3	239.9
	ACS	$\mathfrak{C} \times \mathfrak{C}$	74.8	140.8	251.2	66.7	134.9	239.3
	ACS	$\mathfrak{C} \times P$	38.8	44.3	252.9	37.4	47.2	252.4
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	15.2	54.3	237.3	12.2	19.6	225.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	69.2	139.1	255.7	68.7	132.9	238.6
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	52.0	102.9	249.2	31.5	91.2	248.7
whizzkids97-jsp	ACO_{undir}	—	29.5	83.9	176.0	21.8	76.5	175.5
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	10.8	18.0	149.3	6.0	11.1	151.1
	ACS	$\mathfrak{C} \times \mathfrak{C}$	36.3	50.4	225.5	34.3	49.6	230.3
	ACS	$\mathfrak{C} \times P$	12.6	25.3	145.4	15.0	28.5	147.1
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	2.3	12.0	153.2	0	7.5	156.2
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	39.4	113.1	226.8	36.1	111.0	232.4
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	24.3	72.5	162.4	14.1	67.6	153.7

continues...

Table D.4: GSP results (continued).

Instance	Alg.	Pheromone	no η min	no η med	no η max	η min	η med	η max
whizzkids97-gsp	$\text{ACO}_{\text{undir}}$	—	69.9	137.5	241.8	65.5	130.7	237.5
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	44.8	50.5	215.8	38.0	42.0	200.2
	ACS	$\mathfrak{C} \times \mathfrak{C}$	78.7	97.9	360.3	83.2	107.5	346.7
	ACS	$\mathfrak{C} \times P$	48.8	57.8	246.1	52.2	64.6	223.5
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	29.9	40.7	222.2	23.5	33.5	234.8
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	82.7	187.8	342.9	77.0	186.4	331.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	57.1	125.6	246.3	55.0	120.5	227.1
whizzkids97-osp	$\text{ACO}_{\text{undir}}$	—	51.2	119.5	238.3	48.8	115.6	228.0
	ACS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	14.5	26.1	217.4	8.4	12.1	196.6
	ACS	$\mathfrak{C} \times \mathfrak{C}$	49.6	111.9	215.3	45.9	107.7	222.7
	ACS	$\mathfrak{C} \times P$	22.7	24.8	209.5	22.2	27.4	198.2
	$\mathcal{MM}$AS	$\mathfrak{S}^p \times \mathfrak{C}; \mathfrak{C}^2$	0	16.9	205.5	2.6	10.6	212.1
	$\mathcal{MM}$AS	$\mathfrak{C} \times \mathfrak{C}$	50.1	112.7	234.3	43.5	107.9	225.3
	$\mathcal{MM}$AS	$\mathfrak{C} \times P$	30.3	78.4	226.6	20.3	73.4	209.0

Table D.5: QAP results.

	Assign.			no LS			LS		
Instance	order	Alg.	Pheromone	min	med	max	min	med	max
nug12	SFO	ACO_{undir}	—	6.9	40.5	76.8	0	4.8	20.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.8	22.5	68.9	0	0	15.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.4	37.4	74.4	0	4.5	21.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	38.4	74.0	0	1.7	17.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.5	39.1	77.9	0	4.8	20.4
	DRO	ACO_{undir}	—	6.6	40.5	77.2	0	4.8	20.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	33.6	72.7	0	0	19.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	38.4	76.1	0	1.4	18.0
	HF	ACO_{undir}	—	4.8	40.5	75.4	0	4.8	18.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.2	17.6	74.4	0	0	18.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.8	38.4	74.4	0	4.5	20.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	37.4	74.4	0	1.4	18.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.2	39.4	75.4	0	4.5	20.4
tai12a	SFO	ACO_{undir}	—	8.2	39.3	68.0	0	8.9	24.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.0	29.4	65.7	0	2.8	19.5
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.5	34.8	62.3	0	8.2	22.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.4	34.9	65.1	0	0	20.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.9	36.7	66.3	0	8.3	21.7
	DRO	ACO_{undir}	—	9.4	39.3	68.3	0	8.9	24.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.3	36.0	65.0	0	4.4	22.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.6	35.8	67.4	0	0	21.3
	HF	ACO_{undir}	—	9.8	39.3	66.7	0	8.9	22.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.9	33.8	62.5	0	2.1	20.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.8	38.7	67.9	0	8.6	21.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.5	37.2	66.6	0	0	20.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.8	38.8	68.5	0	8.6	21.9
nug15	SFO	ACO_{undir}	—	8.3	37.7	66.1	0	4.2	18.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.9	21.9	62.4	0	0	15.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.0	35.1	66.1	0	4.0	20.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.3	33.7	60.3	0	0.2	19.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.0	36.2	65.7	0	4.0	20.3
	DRO	ACO_{undir}	—	9.2	37.7	65.6	0	4.2	20.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.7	30.8	61.7	0	0	16.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.3	34.6	63.0	0	0	18.8
	HF	ACO_{undir}	—	10.8	37.7	66.4	0	4.2	19.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.1	19.3	61.4	0	0	15.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.7	36.7	65.4	0	3.8	18.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.7	32.0	63.3	0	0	17.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.1	37.4	66.1	0	3.8	19.8
nug20	SFO	ACO_{undir}	—	13.5	32.6	53.3	0	4.1	12.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.7	22.0	48.6	0	0	9.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	11.4	30.5	52.2	0	4.1	13.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.7	26.8	48.8	0	0	10.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	12.1	30.9	53.6	0	4.1	13.6
	DRO	ACO_{undir}	—	13.2	32.6	52.3	0	4.1	12.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	9.2	27.6	49.3	0	1.2	11.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.7	27.6	48.9	0	0	10.4
	HF	ACO_{undir}	—	12.5	32.6	52.1	0	4.1	12.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.5	20.7	50.9	0	0	11.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.1	31.8	52.8	0	4.0	11.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	24.5	51.0	0	0	10.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	11.3	31.8	54.2	0	4.0	11.9

continues. . .

Table D.5: QAP results (continued).

Instance	Assign. order	Alg.	Pheromone	no LS min	no LS med	no LS max	LS min	LS med	LS max
tai25a	SFO	ACO_{undir}	—	7.2	15.5	24.2	0.1	5.6	11.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.0	13.7	22.9	0	5.3	10.3
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.6	15.0	23.9	0	5.6	10.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.6	12.7	23.1	0.4	1.7	9.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.5	14.8	22.5	0.1	5.6	11.3
	DRO	ACO_{undir}	—	7.3	15.5	24.0	0	5.6	10.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.5	15.1	23.8	0	5.3	10.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.8	14.4	23.8	0.4	1.6	10.2
	HF	ACO_{undir}	—	6.6	15.5	23.4	0.4	5.6	10.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.5	14.0	23.0	0.4	5.2	10.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.9	15.5	23.4	0.4	5.6	11.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.1	13.1	23.3	0.4	1.5	9.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.9	15.5	24.1	0	5.5	11.0
nug30	SFO	ACO_{undir}	—	16.7	32.8	48.8	0	4.1	11.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.5	23.1	46.5	0	0.6	10.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.5	31.1	48.0	0	4.1	12.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.4	25.3	46.5	0	0.3	10.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.5	31.0	48.0	0	4.1	11.7
	DRO	ACO_{undir}	—	16.1	32.8	48.0	0	4.1	11.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.3	28.5	44.5	0	0.7	9.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.2	27.1	44.9	0	0.3	9.3
	HF	ACO_{undir}	—	17.6	32.8	48.4	0	4.1	11.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.5	20.9	45.9	0	0.1	10.3
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	16.8	32.2	47.9	0	4.0	12.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.7	22.0	45.6	0	0.1	9.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	16.3	31.7	50.0	0	4.0	11.8
tai35a	SFO	ACO_{undir}	—	14.0	21.1	28.1	1.2	5.1	9.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.4	19.8	28.8	1.1	4.9	8.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.3	21.1	29.2	1.2	5.1	9.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.5	17.2	27.9	0	1.1	8.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	12.8	21.1	28.8	0.9	5.1	9.1
	DRO	ACO_{undir}	—	13.2	21.1	28.6	1.2	5.1	9.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.7	20.7	28.4	0.9	4.9	8.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.4	19.6	27.7	0.5	1.3	8.7
	HF	ACO_{undir}	—	14.1	21.1	28.5	0.9	5.1	9.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.6	19.8	28.2	0.4	5.0	9.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.0	21.1	28.5	0.8	5.1	9.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.0	17.9	27.3	0.3	1.2	8.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.7	21.1	28.5	1.1	5.1	9.2
ste36a	SFO	ACO_{undir}	—	62.5	137.4	231.3			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	36.3	42.6	203.8			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	22.8	39.0	222.7			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	28.2	93.2	219.6			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	44.9	114.9	231.3			
	DRO	ACO_{undir}	—	66.6	137.4	228.6			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.4	18.3	209.7			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	38.1	95.5	207.9			
	HF	ACO_{undir}	—	64.3	137.4	237.6			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	30.5	44.4	204.2			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	35.4	49.4	228.2			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	28.2	84.2	214.6			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	63.9	131.1	224.1			

continues...

Table D.5: QAP results (continued).

Instance	Assign. order	Alg.	Pheromone	no LS min	no LS med	no LS max	LS min	LS med	LS max
tho40	SFO	ACO_{undir}	—	24.1	42.0	63.0			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.7	24.4	54.4			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	22.4	39.9	61.1			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.1	31.1	59.7			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	21.5	39.9	58.2			
	DRO	ACO_{undir}	—	23.9	42.0	60.5			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	17.8	36.0	56.4			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	16.3	34.4	54.7			
	HF	ACO_{undir}	—	23.1	42.0	59.0			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.6	19.0	56.6			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	23.9	41.1	58.8			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.9	26.4	56.9			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	23.2	40.5	59.9			
sko49	SFO	ACO_{undir}	—	15.4	24.2	33.0			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.3	20.2	31.8			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.0	23.5	32.5			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.8	18.2	30.5			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.0	23.2	31.6			
	DRO	ACO_{undir}	—	16.1	24.2	32.7			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.1	22.3	31.7			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.5	20.5	31.1			
	HF	ACO_{undir}	—	15.6	24.2	32.9			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.5	18.4	31.1			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.9	24.0	32.5			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.9	14.8	33.0			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.0	23.8	32.0			
tai50a	SFO	ACO_{undir}	—	14.3	19.5	24.7			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.6	18.5	24.1			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.9	19.5	25.1			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.3	15.4	23.8			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.1	19.5	25.6			
	DRO	ACO_{undir}	—	13.6	19.5	25.7			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.2	19.2	24.5			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.4	18.2	23.9			
	HF	ACO_{undir}	—	13.6	19.5	25.1			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.9	18.7	24.3			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.3	19.5	25.0			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.9	15.7	23.8			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.9	19.5	25.0			
sko56	SFO	ACO_{undir}	—	16.4	23.8	31.3			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.0	20.4	30.0			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.8	23.1	31.9			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.9	17.7	29.6			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.8	22.9	30.7			
	DRO	ACO_{undir}	—	16.1	23.8	31.3			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.3	22.0	30.4			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.8	19.9	29.4			
	HF	ACO_{undir}	—	16.4	23.8	31.6			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.2	18.3	29.5			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	16.0	23.6	31.8			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.9	14.1	29.5			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.4	23.3	31.0			

continues...

Table D.5: QAP results (continued).

Instance	Assign. order	Alg.	Pheromone	no LS min	no LS med	no LS max	LS min	LS med	LS max
sko64	SFO	ACO_{undir}	—	14.6	21.3	27.8			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.0	18.6	26.7			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.9	20.8	27.3			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.4	15.5	26.3			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.5	20.5	26.7			
	DRO	ACO_{undir}	—	14.7	21.3	27.6			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.6	19.9	26.4			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.9	18.1	26.5			
	HF	ACO_{undir}	—	14.7	21.2	27.6			
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.0	17.0	26.6			
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.7	21.1	27.7			
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.1	12.5	26.2			
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.5	20.9	27.5			

Table D.6: GAP results. Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap1-1	SFO	ACO_{undir}	—	3.3	11.3	18.8	0.3	0	11.9	21.4	3.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.8	4.2	18.5	98.8	3.9	8.3	17.6	99.6
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0.3	11.6	19.3	0.8	1.8	12.2	22.0	6.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	19.0	21.8	0	1.8	20.8	47.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.8	11.9	20.8	1.4	3.3	12.2	22.0	8.2
	DRO	ACO_{undir}	—	0	11.6	21.4	0.9	0	11.9	22.0	5.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	4.5	20.2	99.1	0	3.0	17.6	99.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.6	20.2	37.9	0	0.6	19.9	57.5
	SMC	ACO_{undir}	—	1.8	11.0	21.1	1.4	0.3	11.9	22.3	8.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.3	6.3	17.6	99.5	0.6	3.6	19.0	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0	11.6	19.9	7.7	0.3	11.9	19.9	27.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.5	20.2	49.6	0	3.3	21.4	74.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.8	11.9	20.5	10.5	0.9	12.5	20.5	34.2
	DMC	ACO_{undir}	—	2.1	11.3	22.3	1.4	0.3	12.2	22.3	8.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.8	4.8	16.7	99.6	1.8	6.0	17.3	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	19.9	52.5	0	3.3	20.2	73.9
	DCS	ACO_{undir}	—	1.8	11.3	20.5	1.0	0	11.9	22.3	6.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	4.5	18.8	99.4	0	6.0	17.3	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.0	22.0	44.6	0	1.8	19.6	64.0
	DPS	ACO_{undir}	—	1.8	11.6	22.0	1.2	1.2	11.9	22.3	7.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.0	18.8	99.3	0	0.6	18.8	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.3	21.4	43.8	0	0.6	20.2	63.7
	DPD	ACO_{undir}	—	1.8	11.6	20.5	1.1	0.3	11.9	22.3	6.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.0	16.1	99.2	0	2.7	16.7	99.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	22.3	44.0	0	0.9	20.8	62.8
gap1-2	SFO	ACO_{undir}	—	0.9	7.5	15.0	0*	0	8.3	17.4	0.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.4	15.0	74.5	0.9	2.8	13.8	99.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	2.8	8.0	15.0	0*	1.2	8.6	17.4	2.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	7.5	15.0	0*	0	0	17.7	38.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.7	7.6	15.0	0*	1.5	8.0	17.1	0.1
	DRO	ACO_{undir}	—	0.3	8.6	16.2	0.1	0.6	9.2	17.4	1.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	2.1	16.2	94.3	0.6	1.2	14.4	99.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	3.7	17.1	0.5	0	0.9	17.7	47.9
	SMC	ACO_{undir}	—	0.3	10.1	17.7	0.5	0	9.5	17.7	2.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	7.0	15.6	98.8	0.6	3.4	17.4	99.6
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0	10.4	17.7	2.9	0.6	9.2	17.7	11.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	0.9	17.7	34.8	0	0.9	17.4	62.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0.6	9.8	17.7	3.3	0	7.6	17.7	13.1
	DMC	ACO_{undir}	—	0.3	9.8	17.7	0.4	0	9.5	17.7	2.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	6.4	14.4	99.0	0	3.4	15.0	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	17.1	35.5	0	0.9	17.4	62.7
	DCS	ACO_{undir}	—	1.2	9.5	17.7	0.1	0.6	9.5	17.4	1.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	2.4	14.4	96.7	0	1.8	14.4	99.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.4	9.5	17.7	0.1	0	0.9	17.4	58.2
	DPS	ACO_{undir}	—	0	9.2	16.2	0.2	0.3	9.2	17.7	1.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	2.1	15.6	98.1	0.3	0.9	15.6	99.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	17.1	9.5	0	0.9	17.7	52.8
	DPD	ACO_{undir}	—	1.5	9.2	17.7	0.2	0	9.2	17.7	1.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	2.1	14.1	96.4	0.3	0.9	17.1	99.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	17.4	4.8	0	0.9	17.7	53.1

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap1-3	SFO	ACO_{undir}	—	0	12.1	22.1	1.5	0.3	13.0	23.9	7.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.2	5.3	18.6	99.4	2.9	4.4	22.4	99.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	2.9	13.0	23.6	4.4	1.2	14.2	23.6	11.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	21.8	42.8	0	1.2	22.1	55.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	2.4	13.0	23.6	5.5	1.2	13.9	24.2	13.4
	DRO	ACO_{undir}	—	0.9	13.3	24.2	1.6	0	13.3	23.9	10.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	19.8	99.2	0	2.7	21.2	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	22.1	41.5	0	0.3	21.8	56.1
	SMC	ACO_{undir}	—	1.5	13.0	23.9	1.0	0	13.6	24.5	13.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.4	6.2	23.0	99.5	1.2	4.7	21.2	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.2	11.8	23.6	4.2	0	13.3	24.5	23.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	23.6	40.7	0	2.9	22.1	67.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.5	11.8	23.0	4.3	1.2	12.7	23.6	26.7
	DMC	ACO_{undir}	—	0	13.3	23.3	1.6	0	13.6	24.5	18.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.2	5.6	22.1	99.6	3.8	6.2	23.0	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	21.8	49.1	0	3.5	23.3	73.8
	DCS	ACO_{undir}	—	1.2	13.3	23.9	1.2	0	13.6	24.5	12.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	3.8	19.5	99.5	0	5.0	20.1	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	23.6	41.3	0	2.4	21.2	65.9
	DPS	ACO_{undir}	—	2.1	13.3	23.9	1.8	0	13.3	24.5	12.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	19.5	99.2	0	1.2	20.1	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.5	22.7	42.8	0	1.8	22.1	61.5
	DPD	ACO_{undir}	—	2.1	13.3	23.0	1.7	0.9	13.3	24.2	11.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	2.4	20.1	99.3	1.5	3.5	21.8	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.5	22.7	41.5	0	0.3	22.4	59.6
gap1-4	SFO	ACO_{undir}	—	0	10.0	17.9	0.3	0.9	10.3	19.6	3.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.2	5.0	16.1	98.2	1.8	4.7	15.8	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0	10.9	19.6	1.0	2.1	11.4	19.6	9.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	17.9	23.2	0	0.3	17.9	57.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.5	11.1	19.6	0.9	0.9	10.9	19.6	10.6
	DRO	ACO_{undir}	—	1.5	10.3	18.5	0.8	0.3	10.3	19.6	5.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.8	14.7	99.2	0.3	2.1	15.5	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.3	17.9	36.3	0	0.3	18.8	57.4
	SMC	ACO_{undir}	—	1.2	10.9	19.6	4.4	0.9	10.6	19.6	14.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.9	5.6	17.6	99.7	1.2	2.9	15.8	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0	10.6	19.6	14.5	0.3	10.9	19.1	27.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.6	17.9	51.1	0	2.6	17.9	64.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.2	9.4	19.4	20.1	1.8	10.9	19.6	25.2
	DMC	ACO_{undir}	—	1.2	10.9	19.6	4.3	0.3	10.6	19.6	13.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.3	6.7	17.6	99.7	2.1	4.4	17.9	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.6	18.5	52.0	0	2.6	19.6	65.6
	DCS	ACO_{undir}	—	0	10.6	19.6	2.6	0	10.6	19.6	11.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.9	17.0	99.5	0	2.3	17.6	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	17.6	48.4	0	0.3	18.2	65.1
	DPS	ACO_{undir}	—	0.3	10.6	19.6	1.5	0.3	10.6	19.6	8.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.6	16.4	99.3	0.3	1.8	18.5	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	17.6	41.9	0	0.3	17.0	59.2
	DPD	ACO_{undir}	—	1.2	10.6	19.4	1.5	0	10.6	19.6	8.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	17.9	99.1	0	2.1	17.0	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	18.5	41.1	0	0.3	18.2	57.8

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap1-5	SFO	ACO_{undir}	—	1.2	9.8	18.7	0.9	0.6	10.1	20.2	12.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.5	4.0	18.4	99.2	1.5	4.3	16.9	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.8	9.8	19.0	4.0	1.2	9.5	20.6	16.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.5	18.4	37.1	0	2.1	19.9	64.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0.9	9.2	19.0	2.6	1.2	10.1	20.2	19.8
	DRO	ACO_{undir}	—	0.3	9.8	21.2	2.3	0	10.1	21.5	16.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	1.8	16.0	99.4	0	2.5	19.3	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	19.0	46.3	0	1.8	19.3	67.3
	SMC	ACO_{undir}	—	0	9.8	23.0	10.6	0	10.1	23.0	32.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.5	2.5	16.6	99.8	0.6	2.1	15.6	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.2	7.1	21.2	18.0	0.3	8.0	19.3	34.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	18.7	61.1	0.6	2.1	19.9	76.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	0.6	7.4	21.5	17.9	0	7.7	21.5	37.8
	DMC	ACO_{undir}	—	0	9.8	23.0	11.0	0	10.1	23.0	33.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	3.1	15.6	99.8	0.9	3.1	16.0	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	1.5	19.0	62.9	0	2.5	19.3	81.0
	DCS	ACO_{undir}	—	0.9	9.8	20.9	3.0	0.6	10.1	21.5	20.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	2.1	16.9	99.5	1.8	2.8	16.3	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	19.9	51.6	0	2.8	19.0	75.9
	DPS	ACO_{undir}	—	0.3	9.8	21.5	3.7	0	10.1	21.5	19.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	1.5	19.0	99.4	0.6	1.2	20.6	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	21.2	53.6	0	1.8	19.3	69.8
	DPD	ACO_{undir}	—	0.6	9.8	21.2	3.4	0.3	10.1	21.8	18.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	3.1	16.9	99.5	0.6	1.5	15.6	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	18.7	52.3	0	1.5	18.7	70.2
gap2-1	SFO	ACO_{undir}	—	4.4	17.7	29.3	0.5	5.1	18.0	31.1	8.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.4	7.6	25.3	98.8	3.2	6.2	30.0	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.8	18.0	28.6	0.8	3.2	17.5	30.6	11.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	3.0	30.2	49.4	0.2	2.3	30.0	76.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.5	19.4	30.4	1.8	3.7	15.9	33.4	13.9
	DRO	ACO_{undir}	—	5.3	18.0	30.4	0.6	4.4	18.0	30.4	8.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.8	26.7	98.9	0	1.2	24.2	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.5	30.2	49.3	0	1.2	29.0	73.5
	SMC	ACO_{undir}	—	4.6	18.2	30.0	1.2	4.6	18.0	32.0	14.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	7.8	24.9	99.4	3.0	6.9	25.8	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.5	18.9	33.2	2.2	2.5	18.0	31.6	20.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	28.6	57.9	0	2.5	29.0	79.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.7	18.9	30.0	3.3	3.0	16.8	31.1	20.5
	DMC	ACO_{undir}	—	3.9	18.2	30.0	1.1	3.0	18.0	32.3	15.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	6.5	25.8	99.5	1.2	5.1	25.1	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.6	27.4	55.1	0	1.4	30.9	78.5
	DCS	ACO_{undir}	—	6.7	18.0	30.2	0.9	2.1	17.7	32.9	13.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	4.8	25.3	99.3	0.5	4.8	27.2	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	31.1	51.9	0	1.2	30.2	80.9
	DPS	ACO_{undir}	—	5.5	18.2	31.1	0.9	2.1	18.0	32.0	11.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.8	27.6	99.1	0	0.5	24.7	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.5	29.5	51.7	0	1.2	29.5	76.9
	DPD	ACO_{undir}	—	5.8	18.0	31.3	0.9	3.5	18.0	32.0	11.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	2.5	27.4	98.9	0	1.8	25.6	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.5	30.9	49.3	0	1.2	27.4	74.5

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap2-2	SFO	ACO_{undir}	—	6.7	18.8	33.5	0.7	5.0	18.8	32.8	7.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.0	7.3	30.0	99.2	1.8	4.4	25.2	99.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.0	19.5	31.9	1.5	4.8	19.0	35.1	11.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.2	30.5	50.1	0.5	3.2	30.5	73.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.5	17.9	33.3	2.9	6.2	17.4	31.4	14.5
	DRO	ACO_{undir}	—	5.7	18.8	31.2	0.7	3.9	18.8	35.3	8.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	2.5	27.3	98.6	1.1	3.0	25.2	99.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.5	31.0	49.5	0	0.5	29.8	66.8
	SMC	ACO_{undir}	—	4.8	17.9	31.0	1.2	3.9	18.3	34.4	13.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	6.4	29.8	99.2	2.3	6.4	26.6	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.0	16.5	31.9	2.7	2.5	16.5	32.6	23.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.2	29.4	56.3	0.2	2.5	31.7	80.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.4	15.8	30.7	5.1	0.9	15.4	31.0	26.0
	DMC	ACO_{undir}	—	5.0	17.9	31.2	1.2	3.0	18.3	34.6	13.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.1	5.7	28.2	99.1	3.7	5.3	27.8	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	3.4	31.0	52.5	0.5	2.3	30.5	80.9
	DCS	ACO_{undir}	—	6.9	18.1	30.3	0.7	4.4	18.3	34.6	10.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.5	3.7	27.8	99.0	1.1	4.4	27.3	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	29.8	45.5	0	2.1	29.1	74.6
	DPS	ACO_{undir}	—	5.7	18.6	37.2	0.9	4.8	18.6	33.5	10.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.5	28.0	98.6	0.9	2.8	28.9	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0	30.5	49.1	0	0.5	29.8	68.1
	DPD	ACO_{undir}	—	6.2	18.6	35.3	0.9	4.6	18.6	33.9	10.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	26.6	98.6	0	3.0	25.5	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	29.4	48.6	0	1.8	29.1	69.8
gap2-3	SFO	ACO_{undir}	—	4.5	18.6	33.6	1.6	3.1	18.6	36.7	23.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	6.4	27.1	99.4	2.4	7.4	27.4	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.0	18.3	32.9	3.0	3.6	18.1	33.6	30.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.7	30.0	55.9	0	1.7	30.0	78.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.2	17.9	31.0	6.3	4.0	17.4	33.1	35.0
	DRO	ACO_{undir}	—	4.3	18.6	31.9	2.6	3.6	18.3	34.3	26.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.7	28.1	99.1	0	1.4	29.0	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	31.4	60.0	0	1.7	29.5	79.3
	SMC	ACO_{undir}	—	4.8	18.1	32.4	3.0	2.1	18.1	35.0	30.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.1	5.5	27.6	99.4	1.4	6.0	23.6	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.2	19.8	32.6	6.1	2.1	17.9	34.5	36.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	29.8	62.8	0	1.2	29.3	81.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.0	20.0	35.7	8.9	1.0	18.3	33.6	44.3
	DMC	ACO_{undir}	—	3.6	17.9	31.7	3.4	2.9	18.1	34.3	31.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.1	4.5	25.2	99.4	2.6	5.7	28.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	28.6	65.9	0	1.7	30.2	80.9
	DCS	ACO_{undir}	—	6.2	18.8	31.7	1.5	3.3	18.3	35.0	25.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	4.0	26.2	99.3	1.2	3.8	25.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	29.8	53.7	0	0.5	30.0	79.5
	DPS	ACO_{undir}	—	4.8	18.3	32.4	3.0	2.1	18.3	33.6	28.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	27.1	99.2	0.5	1.2	27.9	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.5	30.0	61.0	0	1.0	29.3	79.5
	DPD	ACO_{undir}	—	6.2	18.6	31.9	2.8	2.6	18.3	33.8	26.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.4	24.3	99.2	0.5	2.4	23.6	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	31.4	57.8	0	1.2	30.5	76.6

continues...

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap2-4	SFO	ACO_{undir}	—	4.5	18.4	30.5	1.2	4.1	18.4	31.5	16.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.3	8.6	26.3	99.1	5.5	6.7	25.8	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.1	17.9	29.6	1.8	4.1	18.1	32.7	19.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.6	29.6	50.6	0	3.1	29.8	78.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.4	16.9	30.3	3.4	3.3	17.4	31.7	21.3
	DRO	ACO_{undir}	—	5.5	18.6	32.9	1.7	4.1	18.4	32.2	19.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	2.6	26.7	99.2	1.7	3.1	27.2	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.9	30.8	57.2	0	2.1	28.4	85.2
	SMC	ACO_{undir}	—	3.8	18.6	32.5	3.4	3.8	18.1	33.2	29.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.0	6.4	27.0	99.5	2.1	3.8	24.3	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.0	18.4	32.7	7.3	2.9	17.7	33.2	39.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	30.1	65.3	0	2.9	28.6	83.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.8	17.9	32.5	8.0	2.1	17.4	32.9	41.3
	DMC	ACO_{undir}	—	5.3	18.9	32.0	3.3	3.1	18.4	32.5	29.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.1	6.7	26.3	99.5	3.8	6.2	26.3	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	30.8	65.4	0	2.1	28.9	81.5
	DCS	ACO_{undir}	—	4.3	18.4	31.5	1.2	4.3	18.4	32.2	19.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.7	5.7	28.4	99.1	1.7	4.5	28.4	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	31.0	53.9	1.0	2.4	29.4	84.8
	DPS	ACO_{undir}	—	5.7	18.6	30.8	2.2	3.8	18.4	32.0	22.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.7	3.1	26.3	99.3	0	2.1	25.8	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.7	30.1	63.0	0	2.1	28.4	80.4
	DPD	ACO_{undir}	—	3.6	18.6	30.5	2.1	3.3	18.4	33.4	20.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	3.1	25.8	99.1	0	2.6	26.5	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	28.9	60.7	1.2	1.7	26.0	81.7
gap2-5	SFO	ACO_{undir}	—	5.1	18.0	31.8	4.9	4.2	17.8	33.4	31.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	4.4	25.0	99.8	2.1	3.5	24.8	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.7	17.3	31.5	8.1	3.3	16.8	31.5	38.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.6	30.4	77.3	0	1.6	28.5	92.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.5	16.8	30.8	9.2	2.6	15.4	31.3	41.7
	DRO	ACO_{undir}	—	4.7	17.8	33.2	3.1	2.6	17.5	32.0	26.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.6	27.8	99.5	0.5	1.9	22.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	28.3	73.8	0	0.9	29.2	89.4
	SMC	ACO_{undir}	—	3.5	18.0	31.1	4.3	2.8	17.8	33.6	34.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.1	5.1	24.8	99.7	1.2	4.7	26.2	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.0	17.3	30.8	7.1	2.3	17.3	32.0	39.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.9	28.7	67.3	0	1.6	29.0	91.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.5	17.5	30.8	8.1	2.6	16.6	32.5	37.7
	DMC	ACO_{undir}	—	3.5	18.0	30.6	4.0	2.6	17.8	34.6	33.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.0	4.4	24.8	99.7	2.3	4.0	29.0	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.1	30.4	72.3	0	1.6	29.0	90.7
	DCS	ACO_{undir}	—	4.2	17.5	31.8	2.9	3.7	17.5	35.7	28.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.6	4.7	26.9	99.5	1.4	4.0	25.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.7	30.4	69.6	0	1.6	26.4	91.8
	DPS	ACO_{undir}	—	5.1	17.8	31.3	3.6	3.0	17.8	33.4	28.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	2.3	31.3	99.5	1.4	2.6	24.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.9	28.7	71.4	0	0.9	29.0	89.1
	DPD	ACO_{undir}	—	4.4	17.8	31.1	3.4	2.1	17.5	32.2	27.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	26.2	99.5	0	1.9	25.0	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.7	28.7	71.4	0	1.2	28.5	90.1

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap3-1	SFO	ACO_{undir}	—	6.2	12.1	17.4	0.1	3.6	12.2	20.7	4.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	5.9	18.1	91.5	2.8	4.0	17.6	99.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.9	12.2	19.0	0.1	3.8	12.1	19.8	6.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.6	12.6	18.1	0.1	0.3	1.9	18.3	77.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.2	13.3	17.4	0*	3.4	11.2	20.9	8.7
	DRO	ACO_{undir}	—	5.0	12.4	20.2	0.2	3.4	12.2	20.5	6.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.9	16.7	97.4	1.0	1.9	17.6	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	19.0	21.1	0	1.4	20.9	75.0
	SMC	ACO_{undir}	—	5.5	12.2	19.3	0.8	3.8	12.2	20.7	13.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.6	5.0	17.2	99.1	2.4	4.1	17.2	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	5.5	12.1	20.3	0.9	2.2	12.1	20.5	15.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	1.4	18.6	62.1	0	2.1	17.8	87.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.8	11.9	19.0	1.4	3.4	11.4	19.5	18.0
	DMC	ACO_{undir}	—	5.9	12.4	20.2	0.8	3.3	12.4	20.9	13.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.7	5.3	17.8	99.2	2.9	4.7	18.8	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.7	18.8	59.0	0	2.1	18.4	88.1
	DCS	ACO_{undir}	—	6.7	12.6	18.8	0.2	4.5	12.4	20.5	8.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.2	4.5	17.6	98.2	3.1	4.3	17.4	99.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.0	19.7	34.1	0	1.2	19.5	80.5
	DPS	ACO_{undir}	—	6.6	12.4	19.3	0.4	4.1	12.4	20.5	9.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.4	16.4	98.5	0.5	2.4	18.3	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	19.5	37.8	0	1.4	18.4	78.4
	DPD	ACO_{undir}	—	5.7	12.2	19.0	0.3	4.1	12.4	20.3	8.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.9	2.1	19.8	98.2	1.0	2.4	16.9	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	19.1	35.7	0.2	1.4	20.7	78.3
gap4-1	SFO	ACO_{undir}	—	9.5	18.1	27.7	0.3	6.1	18.1	29.0	13.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.5	6.9	23.3	98.5	3.8	6.1	24.4	99.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.1	18.0	28.0	0.4	5.2	18.1	28.5	17.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.8	25.3	42.2	0.5	2.0	26.7	87.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.5	17.8	27.4	0.8	6.9	17.8	28.7	20.2
	DRO	ACO_{undir}	—	9.8	17.8	27.9	0.3	7.3	18.0	28.2	10.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.6	23.3	97.0	0.2	1.5	25.0	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.8	26.4	34.6	0	0.8	26.5	83.2
	SMC	ACO_{undir}	—	8.7	18.3	26.4	0.6	5.3	18.3	29.0	16.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.0	6.9	24.8	98.9	4.0	5.2	23.9	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.7	17.8	27.0	0.5	5.2	18.1	29.7	16.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	1.2	26.4	58.5	0	1.1	26.2	84.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.2	18.3	27.4	0.8	6.3	18.1	28.7	18.0
	DMC	ACO_{undir}	—	7.8	18.0	26.7	0.6	7.3	18.3	30.5	17.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.7	8.1	23.5	98.9	4.3	6.1	24.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	1.7	25.3	58.7	0	2.6	25.2	89.9
	DCS	ACO_{undir}	—	9.1	17.5	26.1	0.2	7.0	18.1	29.1	13.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.0	5.2	22.4	96.9	3.4	5.6	24.2	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.4	24.4	29.5	0.2	0.9	24.4	84.8
	DPS	ACO_{undir}	—	8.2	17.8	27.7	0.3	6.3	18.0	28.4	12.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.4	25.0	97.9	0.3	2.3	25.2	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.5	25.8	44.4	0	0.9	25.5	83.7
	DPD	ACO_{undir}	—	9.3	18.0	26.5	0.3	7.2	18.0	29.3	11.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.5	24.1	97.5	0.5	2.7	23.2	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	0.6	27.9	36.8	0	1.1	26.4	82.7

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap5-1	SFO	ACO_{undir}	—	5.5	14.2	22.9	8.4	4.8	14.2	23.4	49.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.7	4.4	20.2	99.6	3.0	4.1	19.4	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	3.7	13.3	22.2	13.8	3.7	13.1	24.2	55.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	21.0	69.0	0	2.7	20.2	89.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.6	13.7	22.6	17.7	2.8	12.4	22.7	57.1
	DRO	ACO_{undir}	—	5.5	14.4	22.7	4.5	4.6	14.4	23.8	40.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	2.1	18.8	99.5	0.7	2.7	19.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	23.3	67.0	0.2	2.1	19.9	92.3
	SMC	ACO_{undir}	—	5.3	15.1	23.3	6.6	5.0	14.6	23.8	49.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.0	4.8	22.2	99.6	3.4	4.1	20.6	100.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	4.4	14.9	25.4	11.9	4.6	14.2	23.6	53.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	21.8	74.7	0	2.5	22.2	96.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.4	14.9	24.0	14.6	3.9	13.9	23.4	57.5
	DMC	ACO_{undir}	—	5.9	15.1	24.5	6.7	4.8	14.6	23.4	49.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.3	4.3	20.1	99.7	2.0	3.2	21.1	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.3	20.8	74.2	0	2.5	21.1	95.6
	DCS	ACO_{undir}	—	6.2	14.6	23.6	4.5	3.6	14.6	23.6	41.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.6	3.2	21.0	99.6	1.8	3.9	19.5	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.6	22.7	69.6	0	2.5	19.7	93.9
	DPS	ACO_{undir}	—	5.5	14.4	22.6	4.8	4.1	14.4	23.8	42.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	1.4	22.0	99.6	1.8	3.0	20.6	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	20.8	65.7	0	2.1	20.8	93.2
	DPD	ACO_{undir}	—	5.9	14.4	22.9	4.3	5.3	14.4	23.1	39.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	2.0	19.0	99.4	1.1	3.0	19.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.6	21.3	68.2	0	2.1	20.1	91.1
gap6-1	SFO	ACO_{undir}	—	7.6	15.0	23.3	0.5	6.3	15.1	24.4	26.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	5.8	19.2	98.8	3.0	5.0	21.0	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.1	15.1	22.7	0.6	6.2	15.1	24.0	26.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	22.7	51.9	0	1.4	21.0	94.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.4	15.2	22.6	0.8	6.8	15.0	24.4	28.4
	DRO	ACO_{undir}	—	7.6	15.4	22.5	0.8	6.4	15.4	24.0	28.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.8	1.8	20.4	99.2	0.4	2.8	20.2	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	21.6	66.5	0.3	1.3	21.4	96.8
	SMC	ACO_{undir}	—	7.5	15.8	23.1	1.1	6.7	15.5	24.7	36.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.7	5.8	22.6	99.4	3.5	5.0	19.1	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.1	15.8	23.3	2.1	5.4	15.0	23.8	40.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.6	22.7	74.0	0.3	1.4	21.9	98.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	6.8	15.6	24.0	2.9	5.0	14.6	24.7	47.8
	DMC	ACO_{undir}	—	8.5	15.8	22.6	1.2	6.6	15.5	25.2	37.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	5.3	20.4	99.4	3.0	5.0	19.1	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.1	1.6	21.8	76.9	0.3	1.4	21.6	99.2
	DCS	ACO_{undir}	—	8.1	15.9	24.0	0.9	6.8	15.5	23.8	35.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.0	3.5	22.5	99.3	2.6	4.1	20.0	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.4	21.7	73.0	0	1.3	21.3	98.4
	DPS	ACO_{undir}	—	8.5	15.5	23.3	1.1	6.8	15.4	24.7	32.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	2.4	20.5	99.3	1.2	3.3	20.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.3	22.2	74.2	0.1	1.4	22.3	97.1
	DPD	ACO_{undir}	—	7.8	15.5	22.9	1.0	7.5	15.4	23.8	30.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	2.2	19.8	99.1	1.4	2.8	20.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.2	22.5	69.7	0.1	1.4	22.5	97.4

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap7-1	SFO	ACO_{undir}	—	9.7	15.5	20.0	0.2	7.4	15.3	23.2	21.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.8	5.8	19.4	97.1	3.7	4.7	19.2	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.7	15.6	20.6	0.2	8.3	15.3	22.7	24.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.6	22.9	20.5	0.3	1.3	21.0	92.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.0	14.9	21.2	0.2	7.6	15.2	23.4	30.4
	DRO	ACO_{undir}	—	10.1	15.2	20.4	0.2	7.2	15.1	22.6	23.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.6	20.9	97.0	1.7	3.7	18.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.3	21.1	34.2	0.3	1.5	21.0	94.6
	SMC	ACO_{undir}	—	9.1	15.2	20.6	0.4	6.5	15.0	22.8	31.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.2	5.6	19.9	98.6	4.4	5.3	18.0	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.9	15.1	22.0	0.4	6.9	14.8	22.8	33.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.9	20.2	57.5	0.3	1.5	20.6	98.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.6	14.9	20.8	0.5	7.4	14.3	21.7	39.9
	DMC	ACO_{undir}	—	8.5	15.0	21.1	0.3	6.3	15.0	22.9	31.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.7	5.6	20.1	98.3	3.7	5.0	18.8	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.1	22.6	54.7	0.2	1.4	21.5	97.6
	DCS	ACO_{undir}	—	9.2	14.8	21.4	0.3	7.5	15.0	22.7	26.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.7	3.4	18.4	98.5	3.9	4.5	19.1	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.1	1.3	20.4	33.6	0.2	1.4	20.0	96.3
	DPS	ACO_{undir}	—	8.8	15.1	21.7	0.3	6.6	15.1	22.6	26.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	1.4	18.4	98.6	1.9	3.8	18.6	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.1	1.1	20.5	53.4	0.2	1.6	20.0	95.6
	DPD	ACO_{undir}	—	9.8	15.2	21.0	0.3	7.1	15.1	23.2	24.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	1.6	20.8	98.7	2.0	3.9	19.5	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.2	1.3	21.2	42.9	0.2	1.4	21.0	95.3
gap8-1	SFO	ACO_{undir}	—	cns	cns	cns	0	8.0	14.2	20.2	0.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	4.3	5.9	18.0	96.8
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	cns	cns	cns	0	9.7	14.4	19.9	0.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	0.9	2.0	19.2	29.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	cns	cns	cns	0	9.2	14.5	20.0	0.2
	DRO	ACO_{undir}	—	14.7	15.8	16.8	0*	10.1	14.6	18.6	0.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.7	15.8	16.8	0*	0.8	1.3	17.8	96.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	14.7	15.8	16.8	0*	0.7	2.7	19.3	11.4
	SMC	ACO_{undir}	—	cns	cns	cns	0	9.2	14.3	19.9	0.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	4.6	5.5	18.4	99.2
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	cns	cns	cns	0	8.4	14.3	18.9	0.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	1.1	1.9	19.9	53.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	cns	cns	cns	0	8.9	14.6	20.4	0.6
	DMC	ACO_{undir}	—	cns	cns	cns	0	9.8	14.3	20.1	0.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	3.6	5.8	17.4	98.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	0.6	2.0	19.2	52.7
	DCS	ACO_{undir}	—	cns	cns	cns	0	9.2	14.3	19.8	0.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	3.4	4.9	19.9	97.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	0.8	1.9	19.8	32.6
	DPS	ACO_{undir}	—	cns	cns	cns	0	10.5	14.5	19.1	0.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	1.0	1.3	18.8	96.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	cns	cns	cns	0	0.4	1.7	20.2	30.3
	DPD	ACO_{undir}	—	13.3	13.3	13.3	0*	10.0	14.4	18.9	0.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.3	13.3	13.3	0*	0.7	1.6	18.0	96.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.3	13.3	13.3	0*	0.6	2.6	19.5	16.1

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap9-1	SFO	ACO_{undir}	—	8.9	16.8	23.7	1.6	7.3	16.8	26.2	28.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.5	5.8	23.7	99.3	3.1	5.6	21.0	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	7.9	15.9	24.5	3.5	6.5	15.7	25.0	33.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.4	23.0	57.9	0.3	2.0	25.2	89.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.0	16.2	23.6	4.0	7.1	15.5	24.5	37.8
	DRO	ACO_{undir}	—	9.2	16.5	24.4	1.1	7.6	16.6	25.7	25.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	2.0	21.6	99.1	1.6	3.7	22.6	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	23.1	62.2	0.3	2.1	23.4	90.7
	SMC	ACO_{undir}	—	8.2	16.6	24.3	2.0	7.2	16.8	25.7	34.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.5	6.3	20.7	99.1	3.2	5.6	20.9	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	8.7	16.9	27.4	3.1	7.2	16.2	25.1	35.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.8	23.0	72.3	0.1	2.4	22.6	97.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.6	16.8	24.3	3.9	6.5	15.9	25.5	41.3
	DMC	ACO_{undir}	—	9.4	16.6	23.8	1.9	7.2	16.6	26.0	34.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.8	6.6	22.3	99.4	3.7	6.1	20.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	2.4	22.7	68.9	0.1	2.4	22.8	97.3
	DCS	ACO_{undir}	—	7.2	16.6	24.0	0.6	6.8	16.6	25.5	21.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	3.7	22.1	98.9	1.7	5.1	22.0	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.3	22.8	56.5	0.3	2.1	24.0	92.1
	DPS	ACO_{undir}	—	7.9	16.5	24.0	1.2	6.9	16.5	25.2	26.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.6	2.1	21.0	99.2	2.0	3.7	20.6	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.1	23.3	67.6	0.1	2.1	22.6	92.0
	DPD	ACO_{undir}	—	8.3	16.6	24.0	1.1	6.6	16.6	26.0	24.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.3	2.1	20.9	99.2	2.4	4.4	20.2	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.8	24.1	64.3	0.3	2.1	23.0	91.9
gap10-1	SFO	ACO_{undir}	—	10.2	16.5	22.2	0.3	8.7	16.4	23.8	21.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.1	5.9	20.8	98.0	4.2	5.1	20.8	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	10.6	16.5	22.8	0.3	8.7	16.2	24.4	21.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	2.3	23.5	38.9	0.2	1.6	21.4	97.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	10.2	16.4	22.7	0.3	8.2	16.1	24.7	23.6
	DRO	ACO_{undir}	—	10.6	16.5	21.9	0.2	7.7	16.3	24.5	17.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.5	2.5	21.2	96.7	2.6	4.1	21.3	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.2	21.8	28.2	0.1	1.5	22.8	94.7
	SMC	ACO_{undir}	—	10.0	16.4	21.5	0.4	7.6	16.4	24.0	28.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.3	6.4	21.3	98.7	3.5	5.3	20.6	100.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	10.3	16.3	22.5	0.6	8.0	16.0	24.3	32.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.1	23.2	58.4	0.2	1.6	21.6	99.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	9.3	16.2	23.0	0.9	8.5	15.8	23.5	38.4
	DMC	ACO_{undir}	—	10.1	16.5	22.1	0.5	8.6	16.3	23.8	28.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.5	5.0	21.0	98.5	4.4	5.1	20.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.3	2.3	22.1	57.9	0.3	1.7	22.0	99.1
	DCS	ACO_{undir}	—	9.5	16.4	21.9	0.2	7.7	16.3	24.0	23.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.6	5.3	21.7	98.6	3.8	4.6	20.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	2.6	21.3	28.3	0.2	1.5	21.9	97.5
	DPS	ACO_{undir}	—	10.5	16.3	21.6	0.3	8.2	16.3	23.7	21.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.8	2.5	21.1	98.3	3.5	5.0	21.9	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.1	1.8	23.2	46.2	0.2	1.5	21.6	97.5
	DPD	ACO_{undir}	—	10.6	16.5	22.4	0.3	8.5	16.3	24.4	19.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	2.2	21.7	98.1	2.4	4.5	19.8	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	1.9	21.7	36.1	0.2	1.4	21.9	97.1

continues. . .

Table D.6: GAP results (continued). Entries of **0*** in the **% feas.** column are values less than 0.05. Entries of **cns** indicate that no feasible solutions were produced.

	Assign.			no η				η			
Instance	order	Alg.	Pheromone	min	med	max	% feas.	min	med	max	% feas.
gap11-1	SFO	ACO_{undir}	—	16.3	24.2	32.4	0.6	14.0	24.3	34.5	37.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.1	9.9	29.5	98.6	5.2	8.8	28.8	100.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	15.9	24.6	32.1	0.8	14.3	24.3	35.3	35.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.6	31.8	58.9	0.3	2.5	32.5	98.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	17.4	24.6	32.2	1.1	14.5	24.2	34.9	39.4
	DRO	ACO_{undir}	—	16.5	24.6	32.9	0.6	14.0	24.4	35.6	38.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.2	3.9	30.7	98.8	5.7	7.2	28.8	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.2	31.6	70.6	0.5	2.5	30.8	99.2
	SMC	ACO_{undir}	—	17.0	24.8	34.8	1.3	14.0	24.4	34.9	50.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.7	8.1	29.6	99.0	6.2	8.0	29.5	100.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	16.6	24.5	33.0	1.6	12.6	24.0	34.5	48.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	3.4	31.5	78.8	0.7	2.7	30.8	99.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	14.0	24.4	33.5	2.6	12.7	23.6	34.8	50.6
	DMC	ACO_{undir}	—	15.4	24.7	32.7	1.3	14.0	24.4	34.1	50.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.7	9.2	30.4	99.2	7.1	8.4	30.7	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.2	3.2	32.8	78.4	0.7	2.6	32.5	99.7
	DCS	ACO_{undir}	—	14.0	24.4	33.5	0.8	13.5	24.3	35.3	45.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.4	6.5	31.8	99.1	6.0	7.7	33.0	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.1	3.0	32.2	72.4	0.4	2.5	33.0	99.6
	DPS	ACO_{undir}	—	15.8	24.6	32.1	0.9	14.5	24.4	35.0	43.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.0	4.2	29.1	99.2	5.4	6.9	28.2	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.4	2.4	32.6	76.9	0.5	2.6	32.9	99.6
	DPD	ACO_{undir}	—	17.0	24.6	33.4	0.8	13.6	24.4	37.1	41.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.6	3.9	30.8	99.0	5.4	7.0	30.6	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.6	2.5	31.4	72.2	0.6	2.5	31.6	99.3
gap12-1	SFO	ACO_{undir}	—	13.3	16.0	19.7	0*	9.9	15.9	22.8	14.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.0	5.3	19.7	27.6	4.4	6.1	20.1	99.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.3	16.0	18.3	0*	9.5	15.7	22.1	13.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.3	15.6	18.7	0*	0.4	1.2	20.7	98.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	13.5	15.2	18.7	0*	9.1	15.5	22.1	14.2
	DRO	ACO_{undir}	—	12.3	15.6	19.1	0*	9.5	15.8	21.8	17.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.5	1.9	19.9	59.4	3.8	5.2	19.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.7	16.6	19.5	0*	0.1	1.1	20.9	97.6
	SMC	ACO_{undir}	—	12.7	15.7	19.3	0*	9.3	15.9	22.4	24.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.5	5.8	20.9	91.3	4.5	5.4	19.0	100.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	12.9	15.6	20.3	0*	9.2	15.7	22.5	24.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.4	15.6	20.2	0*	0.3	1.2	20.3	99.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	12.9	15.7	19.3	0*	9.5	15.4	22.1	26.4
	DMC	ACO_{undir}	—	12.5	16.0	19.7	0*	9.8	15.9	22.5	25.4
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	5.8	19.7	86.0	4.8	5.9	18.8	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.3	15.5	20.9	0*	0.3	1.2	20.3	99.5
	DCS	ACO_{undir}	—	13.0	15.7	19.5	0*	9.4	15.8	21.9	20.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.3	3.7	20.0	75.4	3.8	4.8	18.7	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.5	15.9	18.2	0*	0.2	1.0	20.4	98.5
	DPS	ACO_{undir}	—	12.8	16.1	19.7	0*	9.8	15.9	21.8	21.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.3	2.1	19.2	85.9	3.2	5.0	19.8	100.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.5	15.3	19.2	0*	0.3	1.2	21.2	99.2
	DPD	ACO_{undir}	—	12.7	15.4	18.3	0*	9.2	15.9	22.7	18.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.0	1.6	19.8	78.5	3.8	5.0	20.3	99.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.5	16.1	19.2	0*	0.3	1.2	20.4	99.2

Table D.7: CSeqP results. Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}$; $\mathfrak{C}_{it}^2 \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}$; $\mathfrak{C}_{it}^2$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n20t1	SFO	$\text{ACO}_{\text{undir}}$	—	15.5	160.3	422.4
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	5.2	96.6	317.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	24.1	36.2	305.2
		ACS	$\mathsf{PH}_{\text{same-model}}$	10.3	120.7	337.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	17.2	153.4	422.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	10.3	122.4	360.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.3	37.9	384.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	10.3	122.4	360.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	12.1	146.6	405.2
	DRO	$\text{ACO}_{\text{undir}}$	—	12.1	110.3	351.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	10.3	15.5	270.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.7	13.8	267.2
		ACS	$\mathsf{PH}_{\text{same-model}}$	8.6	96.6	331.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0	87.9	336.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	22.4	294.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	12.1	98.3	348.3
n20t2	SFO	$\text{ACO}_{\text{undir}}$	—	20.0	272.5	695.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	27.5	205.0	585.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	20.0	65.0	532.5
		ACS	$\mathsf{PH}_{\text{same-model}}$	22.5	237.5	585.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	20.0	255.0	695.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	20.0	227.5	647.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.5	60.0	637.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	20.0	227.5	647.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	20.0	252.5	695.0
	DRO	$\text{ACO}_{\text{undir}}$	—	15.0	190.0	620.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	7.5	20.0	462.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.5	30.0	440.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	10.0	155.0	575.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0	122.5	540.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	47.5	525.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	5.0	172.5	617.5
n20t3	SFO	$\text{ACO}_{\text{undir}}$	—	10.3	200.0	469.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	13.8	144.8	424.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.8	41.4	424.1
		ACS	$\mathsf{PH}_{\text{same-model}}$	17.2	151.7	424.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	10.3	182.8	469.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	10.3	165.5	434.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.9	31.0	448.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	10.3	165.5	434.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	10.3	179.3	469.0
	DRO	$\text{ACO}_{\text{undir}}$	—	6.9	131.0	441.4
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	3.4	13.8	313.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.3	17.2	300.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	6.9	113.8	389.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	3.4	89.7	365.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	20.7	365.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	6.9	117.2	406.9

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n20t4	SFO	$\text{ACO}_{\text{undir}}$	—	110.0	750.0	1570.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	90.0	540.0	1310.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	160.0	250.0	1310.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	110.0	570.0	1320.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	110.0	700.0	1570.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	100.0	650.0	1410.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	10.0	150.0	1400.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	100.0	650.0	1410.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	70.0	680.0	1570.0
	DRO	$\text{ACO}_{\text{undir}}$	—	70.0	520.0	1350.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	40.0	80.0	1160.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	60.0	110.0	1160.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	40.0	460.0	1320.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	20.0	480.0	1380.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	120.0	1240.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	50.0	480.0	1280.0
n20t5	SFO	$\text{ACO}_{\text{undir}}$	—	10.7	113.3	252.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	14.7	116.0	240.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.3	16.0	240.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	16.0	100.7	240.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.3	85.3	252.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	1.3	100.7	245.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	16.0	249.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	1.3	100.7	245.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	1.3	85.3	252.0
	DRO	$\text{ACO}_{\text{undir}}$	—	1.3	76.0	242.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	0	12.0	192.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	12.0	163.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	1.3	68.0	242.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0	26.7	243.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	10.7	176.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	1.3	68.0	243.3
n40t1	SFO	$\text{ACO}_{\text{undir}}$	—	27.4	151.4	456.8
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	26.7	122.6	419.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.3	24.7	336.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	26.7	137.7	430.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	26.0	147.9	445.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	26.7	136.3	429.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.8	21.2	377.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	26.7	136.3	429.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	27.4	149.3	452.1
	DRO	$\text{ACO}_{\text{undir}}$	—	20.5	103.4	375.3
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	8.9	11.6	220.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.4	8.2	220.5
		ACS	$\mathsf{PH}_{\text{same-model}}$	21.2	98.6	327.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0.7	18.5	294.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0	8.9	331.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	24.7	98.6	349.3

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n40t2	SFO	$\text{ACO}_{\text{undir}}$	—	60.6	287.2	711.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	60.6	260.6	752.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	35.1	52.1	488.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	57.4	267.0	727.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	73.4	279.8	720.2
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	55.3	267.0	676.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.8	45.7	622.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	55.3	267.0	676.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	72.3	279.8	734.0
	DRO	$\text{ACO}_{\text{undir}}$	—	43.6	197.9	536.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	11.7	28.7	433.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	9.6	23.4	321.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	54.3	190.4	505.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	2.1	43.6	495.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.1	25.5	472.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	34.0	190.4	539.4
n40t3	SFO	$\text{ACO}_{\text{undir}}$	—	43.9	243.9	515.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	45.5	198.5	501.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	19.7	30.3	412.1
		ACS	$\mathsf{PH}_{\text{same-model}}$	39.4	216.7	506.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	43.9	233.3	560.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	45.5	215.2	492.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	9.1	18.2	436.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	45.5	215.2	492.4
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	43.9	231.8	545.5
	DRO	$\text{ACO}_{\text{undir}}$	—	22.7	139.4	407.6
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	9.1	13.6	318.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	7.6	10.6	315.2
		ACS	$\mathsf{PH}_{\text{same-model}}$	22.7	134.8	383.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	4.5	25.8	354.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	4.5	10.6	345.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	19.7	134.8	369.7
n40t4	SFO	$\text{ACO}_{\text{undir}}$	—	130.3	560.6	1233.3
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	90.9	469.7	1127.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	42.4	87.9	875.8
		ACS	$\mathsf{PH}_{\text{same-model}}$	103.0	509.1	1106.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	127.3	536.4	1187.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	121.2	509.1	1078.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.1	57.6	1042.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	121.2	509.1	1078.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	130.3	536.4	1193.9
	DRO	$\text{ACO}_{\text{undir}}$	—	87.9	351.5	897.0
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	12.1	45.5	606.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.1	18.2	560.6
		ACS	$\mathsf{PH}_{\text{same-model}}$	63.6	339.4	957.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0	69.7	769.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.1	33.3	903.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	78.8	339.4	833.3

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n40t5	SFO	$\text{ACO}_{\text{undir}}$	—	26.1	108.2	235.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	29.0	115.3	223.3
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.4	16.5	202.6
		ACS	$\mathsf{PH}_{\text{same-model}}$	20.7	102.3	219.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	23.0	94.9	228.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	20.7	102.8	221.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.3	10.2	202.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	20.7	102.8	221.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	21.6	94.9	226.1
	DRO	$\text{ACO}_{\text{undir}}$	—	15.3	65.6	166.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	2.8	6.3	150.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	4.8	115.9
		ACS	$\mathsf{PH}_{\text{same-model}}$	13.4	63.4	171.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0.6	11.6	148.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.8	8.0	136.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	14.8	63.4	164.8
n60t1	SFO	$\text{ACO}_{\text{undir}}$	—	39.1	145.8	453.4
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	41.6	140.8	540.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	13.0	26.5	336.6
		ACS	$\mathsf{PH}_{\text{same-model}}$	39.1	137.0	406.3
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	39.1	143.7	439.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	36.6	137.4	418.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.3	20.2	336.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	36.6	137.4	418.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	39.1	144.5	437.0
	DRO	$\text{ACO}_{\text{undir}}$	—	34.5	99.6	322.3
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	8.8	11.8	241.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	17.2	67.6	224.4
		ACS	$\mathsf{PH}_{\text{same-model}}$	32.4	96.6	308.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	2.5	13.4	242.0
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.8	7.6	244.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	28.2	96.6	281.5
n60t2	SFO	$\text{ACO}_{\text{undir}}$	—	97.4	280.3	665.8
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	101.3	284.2	789.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	30.3	44.7	532.2
		ACS	$\mathsf{PH}_{\text{same-model}}$	80.3	269.1	653.9
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	93.4	275.7	744.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	77.0	267.8	655.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	23.7	36.8	511.2
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	77.0	267.8	655.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	94.1	276.3	664.5
	DRO	$\text{ACO}_{\text{undir}}$	—	63.8	190.1	441.4
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	18.4	25.7	371.7
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.0	17.1	307.9
		ACS	$\mathsf{PH}_{\text{same-model}}$	50.7	186.8	434.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	5.9	30.3	346.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	17.1	372.4
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	65.8	186.8	433.6

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n60t3	SFO	$\text{ACO}_{\text{undir}}$	—	70.5	249.5	548.6
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	70.5	241.9	581.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	19.0	26.7	371.4
		ACS	$\mathsf{PH}_{\text{same-model}}$	65.7	231.4	526.7
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	70.5	241.9	549.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	72.4	231.4	524.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	5.7	16.2	422.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	72.4	231.4	524.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	69.5	243.8	549.5
	DRO	$\text{ACO}_{\text{undir}}$	—	29.5	139.0	377.1
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	7.6	10.5	297.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	2.9	38.1	297.1
		ACS	$\mathsf{PH}_{\text{same-model}}$	35.2	136.2	378.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	3.8	12.4	301.9
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.0	7.6	308.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	35.2	136.2	380.0
n60t4	SFO	$\text{ACO}_{\text{undir}}$	—	165.5	512.1	1143.1
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	175.9	482.8	1234.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	51.7	67.2	775.9
		ACS	$\mathsf{PH}_{\text{same-model}}$	160.3	475.9	1024.1
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	136.2	500.0	1103.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	155.2	481.0	1062.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	20.7	43.1	910.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	155.2	481.0	1062.1
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	136.2	498.3	1091.4
	DRO	$\text{ACO}_{\text{undir}}$	—	101.7	306.9	801.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	24.1	37.9	644.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.4	160.3	579.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	89.7	301.7	813.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0	32.8	627.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.7	20.7	613.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	106.9	301.7	853.4
n60t5	SFO	$\text{ACO}_{\text{undir}}$	—	27.8	105.0	216.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	34.0	120.1	227.2
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	9.3	13.0	162.8
		ACS	$\mathsf{PH}_{\text{same-model}}$	27.6	101.8	207.3
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	21.5	95.7	216.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	22.8	101.6	209.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.1	11.7	195.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	22.8	101.6	209.8
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	26.5	97.0	216.7
	DRO	$\text{ACO}_{\text{undir}}$	—	19.8	60.5	145.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	2.5	4.6	130.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	13.7	114.9
		ACS	$\mathsf{PH}_{\text{same-model}}$	20.5	59.4	134.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	0.7	8.5	116.7
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	1.1	6.8	117.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	18.1	59.3	147.9

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n80t1	SFO	$\text{ACO}_{\text{undir}}$	—	45.5	143.6	446.7
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	45.8	159.4	600.0
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	17.6	25.5	265.2
		ACS	$\mathsf{PH}_{\text{same-model}}$	38.2	137.9	440.0
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	45.2	142.7	439.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	42.7	138.2	438.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.5	19.7	313.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	42.7	138.2	438.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	44.5	142.1	433.0
	DRO	$\text{ACO}_{\text{undir}}$	—	36.1	97.9	266.1
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	5.5	12.4	226.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	23.9	70.9	176.7
		ACS	$\mathsf{PH}_{\text{same-model}}$	31.5	96.4	255.2
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	4.8	14.2	225.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.9	8.5	211.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	37.9	96.1	262.4
n80t2	SFO	$\text{ACO}_{\text{undir}}$	—	79.1	268.8	586.5
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	103.7	326.0	819.5
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	29.3	39.5	532.1
		ACS	$\mathsf{PH}_{\text{same-model}}$	86.5	261.9	553.5
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	79.1	264.7	578.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	80.0	260.9	591.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.2	34.0	465.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	80.0	260.9	591.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	79.5	266.5	578.1
	DRO	$\text{ACO}_{\text{undir}}$	—	68.8	180.5	431.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	9.8	26.0	322.8
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.5	62.3	273.0
		ACS	$\mathsf{PH}_{\text{same-model}}$	70.7	178.6	415.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	5.6	26.0	316.3
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.5	20.9	300.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	70.7	178.1	454.9
n80t3	SFO	$\text{ACO}_{\text{undir}}$	—	80.1	248.6	549.3
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	71.2	291.1	609.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	12.3	20.5	427.4
		ACS	$\mathsf{PH}_{\text{same-model}}$	53.4	236.3	516.4
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	82.2	243.2	550.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	84.2	237.0	534.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.2	13.0	455.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	84.2	237.0	534.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	82.2	245.2	540.4
	DRO	$\text{ACO}_{\text{undir}}$	—	41.8	134.9	345.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	2.7	7.5	267.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	6.8	74.0	224.7
		ACS	$\mathsf{PH}_{\text{same-model}}$	41.8	133.6	354.1
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	1.4	8.9	334.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.7	6.2	297.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	41.8	133.6	348.6

continues...

Table D.7: CSeqP results (continued). Abbreviations used: $\mathsf{PH}_{\text{assign-pairs}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it} \times \mathfrak{C}_{res}$, $\mathsf{PH}_{\text{same-model}} = \mathfrak{S}^p \times \mathfrak{C}_{it} \times \mathfrak{C}_{res}; \mathfrak{C}^2_{it}$(same model).

Instance	Assign. order	Alg.	Pheromone	min	med	max
n80t4	SFO	$\text{ACO}_{\text{undir}}$	—	203.7	501.2	1074.4
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	176.8	519.5	1365.9
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	36.6	54.9	808.5
		ACS	$\mathsf{PH}_{\text{same-model}}$	180.5	478.0	1046.3
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	202.4	491.5	1070.7
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	191.5	480.5	1034.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	19.5	41.5	900.0
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	191.5	480.5	1034.2
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	178.0	495.1	1057.3
	DRO	$\text{ACO}_{\text{undir}}$	—	120.7	296.3	682.9
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	15.9	29.3	517.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	45.1	176.8	457.3
		ACS	$\mathsf{PH}_{\text{same-model}}$	120.7	292.7	665.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	3.7	29.3	591.5
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.7	17.1	546.3
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	114.6	292.7	659.8
n80t5	SFO	$\text{ACO}_{\text{undir}}$	—	31.5	103.8	203.2
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	36.3	136.7	230.1
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	11.5	13.6	166.1
		ACS	$\mathsf{PH}_{\text{same-model}}$	33.9	101.2	194.6
		ACS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	32.8	97.4	197.5
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	33.9	101.6	194.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	3.6	12.2	175.9
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	33.9	101.6	194.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{res} \times \mathfrak{C}_{res}$	29.7	98.1	197.5
	DRO	$\text{ACO}_{\text{undir}}$	—	22.3	58.3	121.9
		ACS	$\mathsf{PH}_{\text{assign-pairs}}$	0	4.7	105.6
		ACS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	8.4	39.2	105.6
		ACS	$\mathsf{PH}_{\text{same-model}}$	22.0	57.5	125.8
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{assign-pairs}}$	3.1	7.8	105.6
		$\mathcal{MM}$AS	$\mathfrak{C}_{it} \times \mathfrak{C}_{res}$	0.8	4.9	105.6
		$\mathcal{MM}$AS	$\mathsf{PH}_{\text{same-model}}$	22.2	57.4	129.7

Appendix E

Publications Arising from this Study

Publications by the Candidate Relevant to the Thesis

Montgomery, J., Randall, M. and Hendtlass, T. (2004). Search bias in constructive metaheuristics and implications for ant colony optimisation, *in* M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada and T. Stützle (eds), *4th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2004*, Brussels, Belgium, Vol. 3172 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 390–397.

Montgomery, J., Randall, M. and Hendtlass, T. (2005a). Automated selection of appropriate pheromone representations in ant colony optimisation, *Artificial Life* **11**(3): 269–291.

Montgomery, J., Randall, M., and Hendtlass, T. (2005b). Structural advantages for ant colony optimisation inherent in permutation scheduling problems, *in* M. Ali and F. Esposito (eds), *18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2005)*, Bari, Italy, Vol. 3533 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, pp. 218–228.

Additional Publications by the Candidate Relevant to the Thesis but not Forming Part of it

Montgomery, J. and Randall, M. (2002). Anti-pheromone as a tool for better exploration of search space, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms, ANTS2002*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 100–110.

Montgomery, J. and Randall, M. (2003). The accumulated experience ant colony for the traveling salesman problem, *International Journal of Computational Intelligence and Applications* **3**(2): 189–198.

Randall, M. and Montgomery, J. (2002). Candidate set strategies for ant colony optimisation, *in* M. Dorigo, G. Di Caro and M. Sampels (eds), *3rd International Workshop on Ant Algorithms*, Brussels, Belgium, Vol. 2463 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 243–249.

Appendix F

Notation Used in Thesis

Table F.1: Notation used in thesis.

Symbol	Meaning
Constructive heuristics	
$\mathfrak{C}$	set of solution components
$\mathfrak{c}_i \in \mathfrak{C}$	solution component
$\mathfrak{s}$	sequence of solution components $\langle \mathfrak{c}_i, \ldots, \mathfrak{c}_k \rangle$
$\mathfrak{s}^p$	partial sequence of solution components
$\mathfrak{N}(\mathfrak{s}^p)$	set of solution components that may be added to the partial solution $\mathfrak{s}^p$
$\mathcal{A}$	constructive algorithm
$\mathcal{T}_\mathcal{A}$	construction tree defined by $\mathcal{A}$
$\mathfrak{S}$	set of solution sequences
S	set of feasible solutions to a problem
s	a solution
$S_{\mathfrak{s}^p}$	set of solutions which the partial sequence $\mathfrak{s}^p$ could represent once completed
$s_{\mathfrak{s}^p}$	partially completed solution represented by the partial sequence $\mathfrak{s}^p$
$s_\mathfrak{s}$	solution represented by the sequence $\mathfrak{s}$
$\mathfrak{S}_s$	set of sequences corresponding to the solution s
Pheromone representations, miscellaneous symbols	
C	An arbitrary pheromone representation (set of solution characteristics)
$c \in C$	A solution characteristic
Entities that appear in pheromone representations	
$\mathfrak{C}_{it}$	Set of items in an assignment type problem
$\mathfrak{C}_{res}$	Set of resources in an assignment type problem
P	Set of positions of solution components in a sequence
$\mathfrak{S}^p$	Set of partial solutions/sequences