NTNU
Norwegian University of
Science and Technology

Incorporating Labor Division into Ant Colony Optimization

Jørgen Kjeldstad Grimnes
Jakob Hovland

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Keith Downing, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Ant colony optimization (ACO) is a constructivistic and population-based metaheuristic for solving combinatorial optimization problems inspired by how real ants use pheromones to find shortest paths. Like other metaheuristics ACO is prone to converge on local optima, also known as stagnation. Inspired by the collective behavior of real ants, this thesis incorporated labor division into ACO in order to avoid stagnation. Since there was little research that concern utilizing labor division in ACO, two original attempts at merging labor division models from biological science with ACO was performed. The two chosen labor division models were the seemingly popular Fixed-Threshold model and the Self-Reinforcement model.

The two resulting algorithms were applied to the minimum-cost flow problem domain with concave costs functions, which is a NP-hard problem. Max-Min Ant System, also an ACO algorithm, had previously been applied to concave minimum-cost flow problems and achieved good results. The performance and search behavior of the two proposed labor division algorithms were therefore compared to Max-Min Ant System on a set of 300 flow networks. The results indicated that both labor division algorithms performed comparable to Max-Min Ant System and avoided stagnation very well.

Norsk Sammendrag

Denne avhandlingen tok for seg å kombinere ACO-søk med modeller for arbeidsfordeling fra naturen. Et ACO-søk bruker en matematisk modell på hvordan ekte maur samspiller i naturen, eksempelvis for å etablere maurstier. ACO er en konstruktivistisk og populasjonsbasert metaheuristikk som brukes til å løse kombinatoriske optimaliseringsproblem, og har tidligere blitt anvendt til å løse kjente problem slik som Traveling Salesman Problem. I likhet med andre metaheuristikker er ACO utsatt for å konvergere til suboptimale løsninger, også kalt stagnasjon. Etter inspirasjon av maur, kombinerte vi arbeidsfordelingsmønstre man kan observere i naturen med ACO-søk for å forbedre søkeevnen. Mer spesifikt skulle arbeidsfordelingen håndtere balansegangen mellom utforsking av søkelandskapet og å fokusere på å forbedre løsningene.

Siden det var svært lite dokumentert forskning på hvordan biologisk arbeidsfordeling kan bli kombinert med ACO, har vi formulert to nye metoder ved å bruke modeller fra biologiske, empiriske studier. Disse to modellene var Self-Reinforcement og Fixed-Threshold.

De to algoritmene ble testet på et stort testsett av minimum-kost flytproblem med konkave kostfunksjoner, som er en NP-hard problemklasse. De ble også sammenlignet med et annet ACO-søk som tidligere har blitt anvendt til å løse den samme typen problem og har prestert bra.

Eksperimentene utført i sammenheng med avhandlingen viste at arbeidsfordeling fungerer svært bra til å balansere ACO-søk, og oppnår resultater på høyde med den tilsynelatende mest populære strategien for å balansere ACO-søk: Max-Min Ant System.

Preface

This report is the Master's thesis of the authors. The thesis concludes their study in Computer Science at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The report was written in the spring of 2016 with Keith Downing as the project supervisor.

The thesis is related to the field of Swarm Intelligence and Ant Colony Optimization. The reader is assumed to be familiar with both the field of Artificial Intelligence and elementary graph theory.

Jakob Hovland

Jakob Hovland

Trondheim, June, 2016

Jørgen K Grimnes

Jørgen Grimnes

Trondheim, June, 2016

Contents

List of Figures

List of Tables

List of Symbols

α	Pheromone weight. This can be interpreted as the importance of pheromone on an edge. This intuition only holds for $\alpha \geq 1$
β	Visibility weight. This variable can be interpreted as the importance of a high visibility value η on an edge. This intuition only holds for $\beta \geq 1$
η	The edge visibility. The visibility conveying how willingly the ant should traverse the given edge. If an edge has a low visibility, the ant will refrain from choosing the edge unless the edge has a low of pheromones.
γ	Only applicable to FT-ACO. The exploration factor. This constant regulate how willing the scouts will be to follow less traveled edges in the graph. $\gamma \in < 0, 1 >$ where γ equal to 1 imply that the scouts will ignore any difference in pheromone signal.
ρ	Pheromone evaporation rate. $\rho \in < 0, 1 >$
τ_0	Initial pheromone level. All edges in the graph are initialized with a certain pheromone quantity τ_0.
τ_{max}	The upper bounds on the pheromone levels.
τ_{min}	The lower bounds on the pheromone levels.
θ	Exploration Threshold
C_{best}	Cost of the solution S_{best}
C_{OPT}	Cost of the optimal solution S_{OPT}
$c_{ij}(x)$	Cost of the flow on edge $(i, j) \in E$

C_k	Cost of the solution S_k
d_i	Demand of node $i \in N$
E	The set of edges in a flow problem
$G = (N, E)$	Graph composed of nodes N and edges E
N	The set of nodes in a flow problem
S_{best}	The best solution found by a given search.
S_{OPT}	The optimal solution. This is the set of edges $(i, j) \in E$ that constitute the optimal flow path.
S_k	The best solution found in iteration k
x_{ij}	Flow on edge $(i, j) \in E$
K	The number of iterations dictate how many time-steps the algorithm will progress.
M	The number of ants dictate how many solutions that should be generated at each iteration of the search.

List of Acronyms

MCFP	Minimum Cost Flow Problem
SSU MCFP	Single Source Uncapacitated Minimum Cost Flow Problem
ACO	Ant Colony Optimization
MMAS	Max-Min Ant-System
FT-ACO	Fixed-Threshold ACO
SR-ACO	Self-Reinforcement ACO
AS_{rank}	Ranked Ant System
FTM	Fixed Threshold Model
RST	Randomized Spanning Tree

Chapter 1

Introduction

1.1 Research Motivation

Insects have captivated humans for centuries. They are simple creatures on the individual level. Despite that, eusocial insect colonies display complex collective behaviors that far exceed the capabilities of a single individual. Ant colonies are for instance capable of constructing complex mounds and establish the shortest path between their mound and food resources.

Ants have proven to be one of the most successful species on the planet. The first ant walked the earth roughly 100 million years ago. They outlived the dinosaur extinction and witnessed the emergence of the human ancestors. Today, the number of ants is estimated to be approximately 10^{16} individuals [15]. Each ant has the liberty to decide on its own activities. The interplay among a colony of ants forms a self-organized system that is capable of overcoming challenges such as defending their mound and providing food [23]. The self-organization in ant societies is perhaps most evident in the division of labor observed in some ant species [14, 49]. In these societies, the ants tend to specialize towards certain activities within the colony. This can be observed in the genus *Pheidole*, where the larger ants tend to provide security for the mound while the smaller ants forage for food [49].

By combining local decisions and self-organization, the ant colony emerges as a resilient and dynamic system. As an example, when the ratio between larger and smaller workers in *Pheidole* was artificially altered, the colony counteracted the imbalance by workers changing roles [49]. Their resilience enables colonies to endure both internal and external challenges, such as population increase or destruction of their mounds [23]

Thus an ant colony is a collection of simple agents, which combined, is capable of solving complex problems. The emergence of complex behavior from simple agents have inspired the field of Swarm Intelligence to study how mathematical models of ant colonies can be the foundation of algorithms. This has made ant colonies interesting to the field of optimization theory.

Algorithms that solve problems using a model of ants are categorized as Ant Colony Optimization (ACO) algorithms. The goal of ACO is to mimic and adapt the behavior of real ants to solve combinatorial optimization problems. Analogous to real ants, the artificial ants in ACO are only presented with local knowledge of the properties they observe. Thus these algorithms form decentralized systems without any global, controlling entity. ACO algorithms have been applied to a variety of problems such as facility location problems [12], routing problems [39], set cover problems [13], and traveling salesman problems [17, 21].

However, ACO algorithms are prone to converge to suboptimal solutions. In ACO algorithms, converging to a suboptimal solution is also called *stagnation*. The stagnation term was coined by Dorigo et al. [18] and describes a situation where no further solutions would be discovered during a search. Many papers have been published on how to prevent stagnation from occurring during ant search [19, 29, 42, 50]. However, to our knowledge no research has been published on how naturalistic division of labor could be used to avoid this problem.

1.2 Research Topic and Questions

The problem investigated in this paper was the single-source uncapacitated, minimum-cost flow problem (MCFP) with concave cost functions. This is a NP-hard problem [24]. The research topic was:

Incorporating Labor Division into Ant Colony Optimization

The goal of this paper was to evaluate how the two labor division models Fixed-Threshold and Self-Reinforcement model could be combined with the Ant Colony Optimization metaheuristic. The performance of the labor division models was compared to the Max-Min Ant System described by Stützle and Hoos [42], which has previously been applied to the same problem domain by Monteiro et al. [33].

This thesis focused on how labor division can be brought into play to avoid stagnation and balance the search between exploration and exploitation. The following research questions were defined with regard to MCFP problems with concave cost functions:

1 *How can labor division be incorporated into ACO to counter stagnation and how does it affect the performance?*

2 *How are exploration and exploitation in ACO balanced by the Fixed-Threshold and Self-Reinforcement labor division models?*

Exploration and exploitation Balancing exploration and exploitation refer to the decision of whether to invest computational resources in searching untested areas of the search landscape or to search for solutions in proximity of the best solution that has been discovered. In a maritime analogy, this is the dilemma of whether to set all your fishing nets in the same location where you've caught fish before, or try to discover even better spots to fish. While focusing the effort on good regions may lead to small improvements, one are often required to explore the landscape before finding the optimal region. This is when the balance of exploration and exploitation comes into play. While the exploration may lead you toward the optimal region, you are often required to probe the most promising areas before finding the optimal region.

Fixed-Threshold and Self-Reinforcement The Fixed-Threshold Model is a labor division model from biological science capable of describing how a diverse population of agents may specialize to perform different tasks. Its foundation in nature has been advocated by multiple empirical studies, and this has made it a popular model for division of labor. While the Fixed-Threshold Model has a mathematical definition, the Self-Reinforcement model is a concept. The self-reinforcement derives from the anticipated behavior of the individuals. The core hypothesis in the Self-Reinforcement model is that the individuals who perceive some reward after executing a given task will be more likely to execute the same task again than someone who did not experience the same reward. The notion of the Self-Reinforcement model is therefore that division of labor spring from experience rather than genetics.

1.3 Research Method

The principal research method to answer the research questions was to conduct an extensive literature review of modern labor division models. The literature study was used as a basis to devise two labor division schemes for the ACO algorithm, which were tested and compared to the popular Max-Min Ant System algorithm. The result of the literature review is presented in chapter 3.

The computational experiments consisted of applying the three algorithms on a large collection MFCPs, and compare their performance based on statistics. The experiments were intended to identify if the incorporation of labor division could improve the performance of ant-based algorithms in the MFCP domain compared to the Max-Min Ant System algorithm.

1.4 Report Outline

This report is structured accordingly:

Chapter 1 offers an introduction to the problem and presents the research motivation and questions.

Chapter 2 formally defines the minimum cost flow problem with concave cost functions and present background information on the Ant Colony Optimization algorithm.

Chapter 3 presents a literature review. The literature review discusses four themes important to this thesis. First, the chapter provides an introduction to how the realistic test problems used in the comparative were constructed. Second, a review of the research on minimum cost flow problems is introduced. Third, the chapter presents a review of the current countermeasures to stagnation and early convergence in ACO. Fourth, two labor division models are discussed. Finally, the chapter is rounded off with an allusion of related works and a structured literature review.

Chapter 4 details how a solution is constructed. In addition, the chapter presents two combinations of ACO and division of labor models to solve MCFP. The chapter concludes by describing the experimental plan.

Chapter 5 presents and discusses the results

Chapter 6 summarizes the findings and what has been achieved.

Chapter 2

Background

2.1 Introduction

This chapter will briefly introduce the problem considered in this thesis, the minimum-cost flow problem with concave cost functions. A mathematical model of the problem based on graph theory is presented. In addition, we give a short presentation of the ant colony optimization metaheuristic, which serves as a framework for the ACO algorithms presented in this thesis.

2.2 The Minimum-Cost Flow Problem

The minimum-cost flow problem, also known as the transshipment problem, is to find the cheapest way to transport a commodity through a network in order to satisfy demands at some nodes from supplies at other nodes. It is regarded as the most fundamental flow problem [1, p. 4] and arises in many scenarios such as distributing goods from manufacturers to retailers, the flow of parts through a production line, and routing traffic through a road network [1, p. 5].

Formally, the minimum-cost flow problem can be defined as follows. Let

$G = (N, E)$ be a directed graph where N is a set of nodes and E is a set of edges represented as ordered pairs of nodes. Reciprocal edges from nodes to themselves are prohibited. Every node $i \in N$ has an associated supply or demand d_i. If $d_i > 0$ then node i is a supply node, if $d_i < 0$ it follows that node i is a demand node and if $d_i = 0$ then node i is known as a transshipment node. It is required that the supply and demand of all nodes sum to zero, that is $\sum_{i \in N} d_i = 0$. In addition, every edge $(i, j) \in E$ has an associated lower capacity l_{ij}, upper capacity u_{ij} and cost function $c_{ij}(x)$. The objective is to find a flow $X : E \to \mathbb{Z}$ of minimal cost, where x_{ij} denotes the flow through edge (i, j).

$$\text{Minimize} \quad \sum_{(i,j) \in E} c_{ij}(x_{ij}) \tag{2.1}$$

$$\text{subject to} \quad \sum_{j \in N^+(i)} x_{ij} - \sum_{j \in N^-(i)} x_{ji} = d_i, \qquad \forall i \in N \tag{2.2}$$

$$0 \le l_{ij} \le x_{ij} \le u_{ij}, \qquad \forall (i,j) \in E \tag{2.3}$$

Equation 2.2 is known as the flow conservation constraint. It states that the difference in flow leaving and entering a node must be equal to the supply or demand of the node. The first term in Equation 2.2 is the amount of flow leaving node i and the second term is the amount of flow entering node i. Equation 2.3 is known as the capacity constraint and state that the flow through an edge is bounded by the lower and upper capacities of the given edge.

2.2.1 Concave Cost Functions

The complexity of a minimum-cost flow problem is induced by the cost functions used in the network. If the cost functions are linear or convex the problem is solvable in polynomial time. However, if the cost functions are concave the minimum cost flow problem is NP-hard [24], and thus no polynomial time algorithm exists unless $P = NP$. Concave cost functions are often more realistic than linear cost functions and arise in many scenarios due to start-up costs, discounts and economies of scale [24, p. 77]. Yet another reason to solve concave minimum-cost flow problems is that networks with general nonlinear cost functions can be converted into equivalent expanded networks with concave cost functions [28].

A function is said to be concave on an interval X, if it satisfies the following inequality:

$$f(tx_1 + (1-t)x_2) \geq tf(x_1) + (1-t)f(x_2), \quad \forall x_1, x_2 \in X, \forall t \in [0,1] \tag{2.4}$$

Intuitively, this can be understood in the following way. $tx_1 + (1-t)x_2$ is a number between x_1 and x_2 and $tf(x_1) + (1-t)f(x_2)$ is the parametric equation of the straight line between the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$. The definition can thus intuitively be understood as that the point $(tx_1 + (1-t)x_2, f(tx_1 + (1-t)x_2))$ lies above the straight line between the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$. This is illustrated in figure 2.1.

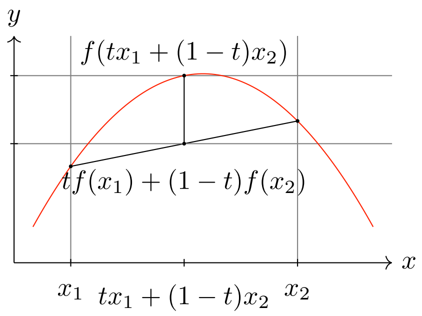
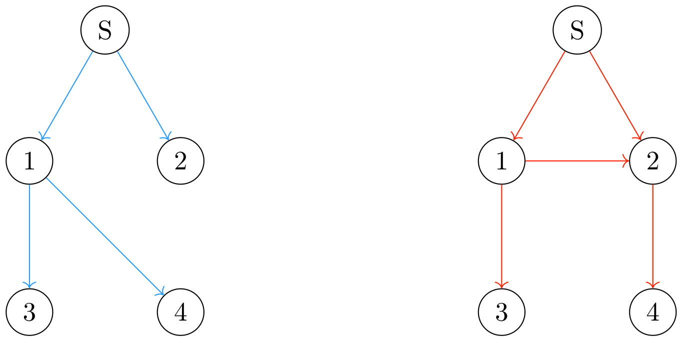


FIGURE 2.1: An example of a concave function. Notice how every point on the curve lies above the straight line from $(x_1, f(x_1))$ to $(x_2, f(x_2))$.

2.2.2 Single-Source Uncapacitated Flow Networks

When dealing with concave cost functions it is common to consider single-source uncapacitated networks, because for such networks the optimal solution to the minimum-cost flow problem is an arborescence rooted at the source node [55]. Knowing that the optimal solution is an arborescence adds additional constraints to the search space and reduces the number of feasible solutions. Yet another reason to consider single-source uncapacitated networks is that there is no loss of generality since multi-source capacitated networks can be converted into equivalent expanded single-source uncapacitated networks.

Arborescences are also known as directed rooted trees or branchings. An illustration of an arborescence can be seen in fig. 2.2. In an arborescence there is exactly one edge entering each node, except the root node which has no incoming edges. Thus, there is exactly one path from the root to every other node.



(a) The blue edges form an arborescence. Notice how every node except the source has exactly one incoming edge.

(b) The red edges does not form an arborescence because node 2 has two incoming edges.

Figure 2.2: Illustration of arborescent flow paths

2.3 Ant Colony Optimization

Ant colony optimization (ACO) is a constructivistic and population-based metaheuristic for solving combinatorial optimization problems inspired by how ants use pheromones to find the shortest path between their nest and a food source. It was first defined by Dorigo and Di Caro [16] and later elaborated by Dorigo and Stützle [19, 20].

The ACO metaheuristic consists of three steps that are repeated until some termination criterion is met. The three steps are: solution construction, daemon actions, and pheromone updating. Typical daemon actions are post-optimization routines such as improving a solution using a local search. While the solution construction and pheromone updating mechanisms are essential in the ACO metaheuristic, daemon actions are optional. Typical termination criteria are either an iteration limit or a convergence measure.

Several special cases of ACO have been proposed in the literature, many predating the ACO metaheuristic. Such special cases are often tailored to specific problems and have well-defined solution construction, daemon actions, and pheromone updating. Some notable special cases are: Ant System [18], Ant Colony System [17], Rank Based Ant System [11], and Max-Min Ant System [42, 43].

2.3.1 Solution Construction

During solution construction a set of M ants construct solutions. Solutions are constructed by repeatedly choosing solution components until the solution is complete. The solution components are chosen according to a probability distribution based on the pheromones. Many different probability distributions have been used by different variations of ACO. A common probability distribution is:

$$p(c_i) = \frac{\tau_i^\alpha \cdot \eta_i^\beta}{\sum\limits_{c_j \in N(s^p)} \tau_j^\alpha \cdot \eta_j^\beta} \tag{2.5}$$

where τ_i is the pheromone associated with solution component c_i, η_i is a heuristic value for solution component c_i, α and β are the weights of the pheromone and heuristic value respectively, s^p is the current partial solution, and $N(s^p)$ denotes the set of possible solution components for the given partial solution.

2.3.2 Pheromone Updating

The pheromones define a probability distribution over the solution space and solution construction is the same as sampling from the probability distribution defined by the pheromones. The purpose of updating the pheromones is to increase the probability of generating good solutions. This is typically achieved by evaporation and depositing pheromones on the solution components in a set of good solutions. Evaporation is inspired by how real pheromone trails erode over time in nature, and gradually diminishes the importance of previously reinforced solutions. A common updating rule is:

$$\tau \leftarrow \rho\tau + \Delta\tau \tag{2.6}$$

where τ is the pheromone, ρ is the evaporation rate and $\Delta\tau$ is the reinforced pheromone distribution. The choice for $\Delta\tau$ varies between different ACO algorithms. For minimization problems, two popular choices are:

$$\Delta\tau = \frac{Q}{S_{\text{best}}} \tag{2.7}$$

where S_{best} is the best solution found at any iteration up until the moment of pheromone updating, and:

$$\Delta\tau = \frac{Q}{S_k} \tag{2.8}$$

where S_k the best solution found during the iteration k. The former being more exploitative and the latter more exploratory. It is also possible to base $\Delta\tau$ on multiple solutions.

2.4 Summary

This chapter has presented the minimum-cost flow problem with concave cost functions and the ACO metaheuristic. A mathematical model for the minimum-cost flow problem was also presented. In addition, concave cost functions and their implications was discussed briefly. Finally, the special properties of single-source uncapacitated problems were specified. Regarding ACO, an overview of how solutions are constructed based on pheromones and how pheromones are updated was presented.

The next chapter presents a literature review that first examines how realistic flow networks can be generated for use in testing. Second, it presents some of the research into concave minimum-cost flow problems. Third, stagnation in ACO is defined and some methods used to avoid stagnation is presented. Finally, an introduction to division of labor and specifically the Fixed-Threshold and Self-Reinforcement models is given.

Chapter 3

Literature Review

3.1 Generating Realistic Test Graphs

The research that concern applying ant algorithms to MCFP appear to only have been tested on graph sets consisting of random graphs. To our knowledge, only one graph set with concave cost functions has been referenced in literature [4]. This set consists of small graph instances with maximum 50 nodes. Unfortunately, random graphs do not model real world transportation networks. Instead, transportation networks pertain to the set of graphs called small-world networks [30, 41, 52]. In a small-world network each node is connected to only a few other nodes, but there should exist paths reaching all other nodes within few jumps.

In addition to having the small-world property, realistic networks tend to be scale-free [30, 41]. A network is scale-free when the edge degree distribution follows a power law. In other words, when there are a few nodes with very many edge connections while the bulk of the nodes are only connected to a few other nodes.

A comparison of a random, small-world and scale free networks is presented in fig. 3.1. It is apparent from the figure that even though the graphs have

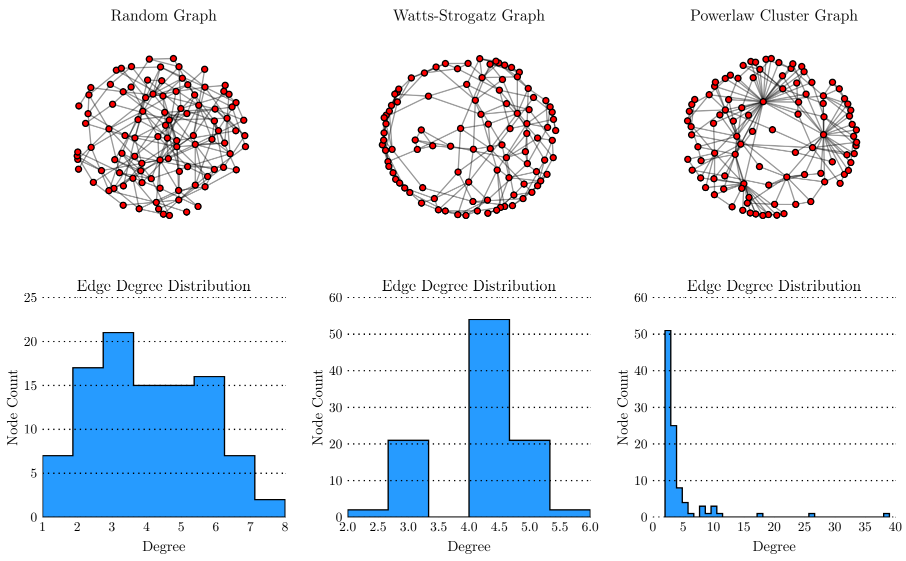


Figure 3.1: A comparison of a random graph (leftmost) and two small-world graphs, where one of them is scale-free (rightmost). The small-world graph in the center was created using the Watts-Strogatz graph generating procedure. The scale-free network on the right, called Powerlaw Cluster Graph, was generated using the procedure formulated by Holme and Kim [25]. It is clear from the figure that the graphs are very different in structure even though the have the same number of nodes and edges. The rightmost graph tend to have the most realistic transportation network structure.

the same number of nodes and edges, they are very different in structure. The rightmost graph is both scale-free and has the small-world property. This graph was generated using a graph growing procedure formulated by Holme and Kim [25]. Similar to the Barabási-Albert model, this procedure generates graphs that abide the power law distribution in addition to ensuring the small-world property [25, 34]. Contrary to the Barabási-Albert model, the procedure by Holme and Kim [25] also produce a high degree of clustering, which is often observed in realistic graphs. Consequently, the small-world graphs in this paper were created using the graph-growing model from Holme and Kim.

The experiment conducted in this thesis tested the algorithms on a graph set composed by 62 small-world graphs in addition to the Beasley graphs [4]. The small-world graphs were included to test the algorithms on more realistic problem instances than the random graphs in the public test set. Furthermore, the test problems were made more realistic by assigning concave cost functions to the flow edges. The next section provides some appreciation for the application of concave cost functions to flow problems and their usefulness.

3.2 The Concave Minimum-Cost Flow Problem

A single-source uncapacitated flow problem with concave cost functions has strong representational power. Any flow problem with non-linear cost functions can be redefined as a flow problem with concave functions in an expanded network [28]. In addition, both capacitated and multi-source networks can be transformed into a single-source uncapacitated problem [46, 55]. Consequently, the domain of single-source uncapacitated minimum-cost flow problems with concave cost functions is worthwhile studying.

When concave cost functions are introduced to the minimum-cost flow problem, the problem become NP-hard [24]. The increase in complexity originates from the curve of the concave functions. A local optimum in a single concave function may not correspond to the global minimum of the system.

A single-source uncapacitated flow problem with concave cost functions has a solution if there are no negative cost cycles and if every demand node is reachable from the source node. If the problem satisfies these two criteria, the solution will be a spanning tree rooted at the source and reaching out to all demand nodes [33, 55]. This analytical property is an important clue when deciding how the ants should construct their solutions in ACO.

Palekar et al. [37] developed a branch and bound algorithm for the fixed charge transportation problem and looked into which properties make certain problems more difficult than others. They observed that the ratio between the total unit costs and fixed charge costs in the optimal solution give an indication to the problems difficulty, with intermediate values being the most difficult. They argued that this could be because if the unit costs are much larger than the fixed charge costs the problem approaches a linear problem and if the fixed charge costs are much larger than the unit costs the problem reduces to minimizing the fixed charges. However, a drawback of this ratio is that it requires the optimal solution and thus cannot be used to estimate the difficulty of the problem before trying to solve it. As a solution, Palekar et al. [37] came up with another way to estimate the difficulty of problems, dubbed Difficulty Ratio (DR), which they defined as:

$$DR = \frac{\overline{f}(m+n-1)}{\overline{c}\sum_{i=1}^{n} d_i} \tag{3.1}$$

where $\overline{f}$ and $\overline{c}$ is the average fixed charge cost and average unit cost of all edges respectively, m is the number of supply nodes, n is the number of demand nodes, and d_i is the demand of node i. This analysis could be used to dynamically tune the parameter configurations of a search algorithm. If the DR indicates that the graph might be difficult, an ACO algorithm could gear its parameters toward exploration rather than exploitation in order to find the optimal solution. When ant algorithms promote exploration, the search tends to postpone the onset of stagnation. The next section explores various stagnation avoidance strategies documented in research. These strategies attempt to provide a mechanism for balancing exploration and exploitation in ACO.

3.3 Stagnation in Ant Colony Optimization

Ant algorithms were first applied to combinatorial optimization problems in 1996 by Dorigo et al. [18]. Dorigo et al. applied their ant algorithm Ant System to the traveling salesman problem (TSP), and achieved good results on small problem instances. However, Ant System did not scale well and could not compete with state-of-the-art optimization algorithms in benchmarking tests [42]. Stützle and Hoos [42] argued that Ant System displayed poor performance because of an excessive focus on exploration. Based on research on combinatorial optimization search landscapes, Stützle and Hoos [42] reasoned that an ant-based algorithm should exploit the search history to a greater extent.

In ant algorithms, early convergence is also known as stagnation. An ant algorithm is said to enter *stagnation behavior* when all ants travel the same suboptimal set of edges [15, 19]. In other words: an ant-based algorithm stagnates when the ants stop exploring new solutions [18].

Pheromone Bounds

Due to pheromone evaporation, the pheromone level on seldom-travelled edges in the graph may approach zero. The likelihood of this edge being travelled is proportional to the pheromone level, and would thus decrease in tandem. When the likelihood of choosing these edges become very low, all of the ants will construct the same solution and thereby have stagnated the search. To guarantee the possibility of traversing neglected edges, it is common to introduce bounds on the edge pheromone level.

Stützle and Hoos [42] introduced both upper and lower bounds on the pheromone strength in their Max-Min Ant System algorithm. The pheromone restriction prevented poor regions of the search space from being completely ignored and good regions from dominating the search, by limiting the pheromones to the range $[\tau_{min}, \tau_{max}]$. Let C_{best} denote the cost of the best solution discovered and ρ be the pheromone evaporation factor. The upper bound was the analytical limit:

$$\tau_{max} = \frac{1}{\rho C_{\text{best}}} \tag{3.2}$$

The minimum bound was dynamically specified by a constant ratio from τ_{max}. Let p_{best} be the probability of constructing the best solution and N be the number nodes in the graph. τ_{min} was defined as:

$$\tau_{min} = \frac{\tau_{max}(1 - p_{best})}{\left(\frac{N}{2} - 1\right) p_{best}} \tag{3.3}$$

The notion of using pheromone bounds to prevent stagnation is appealing both due to its simplicity and effectiveness. The Max-Min Ant System achieved much better results than Ant System [42].

The mathematics behind the pheromone limitation in MMAS implies that the lower bound on the pheromone strength τ_{min} is directly proportional to the upper bound τ_{max}. The ratio between τ_{min} and τ_{max} control the degree of stagnation avoidance. In order to prevent stagnation the ratio $\frac{\tau_{min}}{\tau_{max}}$ must be closer to one, else the edges with τ_{min} pheromone would be impossible to coincidentally choose. However, if the lower bound ratio is close to one the algorithm would yield a very slow convergence rate since the ants will not be able to convey the same amount of information via stigmergy. Contrarily, if the ratio is close to zero the search would be likely to stagnate. Hence, the ratio has a direct impact on the convergence rate of the search.

Altiparmak and Karaoglan [2] and Venables and Moscardini [45] also employed pheromone limitation to counter stagnation and used the same τ_{max} as the Max-Min Ant System. However, they both specified a simpler lower bound τ_{min}. Let a be some problem dependent constant:

$$\tau_{min} = \frac{\tau_{max}}{a} \tag{3.4}$$

The redefinition of τ_{min} is just circumventing the problem of estimating p_{best} required in Max-Min Ant System. Instead, the ratio between τ_{max} and τ_{min} is directly specified.

Bui and Zrncic [10] specified their pheromone bounds to depend on the edge costs in the best solution S_k of iteration k when solving degree-constrained spanning tree problems. In addition, they did not cut off the pheromone values to the respective boundary, but instead either subtracted or added the initial pheromone value τ_0 to the edge in order to validate the pheromone level again. Let C_k^* be the set of costs for the edges in the solution S_k. The maximum and minimum bounds were defined as:

$$\begin{aligned}\tau_{max} &= 1000\left(\max C_k^* - \min C_k^* + \frac{\max C_k^* - \min C_k^*}{3}\right)\\ \tau_{min} &= \frac{\max C_k^* - 3}{3}\end{aligned} \tag{3.5}$$

However, even though limiting the pheromone levels is a popular approach to prevent early convergence it does not cancel stagnation completely. Consequently, many papers suggest to also include a stagnation criterion, that once reached, trigger the pheromone levels to be reset to an arbitrary value τ_0 [6, 19, 29].

Reset Pheromone Levels

Blum and Blesa [6] utilized two stagnation-avoiding strategies when solving the k-cardinality problem. First, they used the pheromones bounds $\tau_{max} = 0.999$ and $\tau_{min} = 0.001$. Finally, they reset both the global solution and the pheromone levels when the system had stagnated. The level of stagnation was described by a convergence factor c_f. Let S_k be the set of edges in the best solution of iteration k and t be the cardinality of the tree, then c_f was defined as:

$$c_f = \frac{\sum_{(i,j)\in S_k} \tau_{(i,j)}}{t \cdot \tau_{max}} \tag{3.6}$$

Following the definition in eq. (3.6), the convergence factor c_f is range bound to $[0, 1]$. The more likely the colony is to construct solution S_k again, the closer will c_f be to one. This is an intuitive measurement for stagnation, since stagnation is defined as when the ants are guaranteed to construct the same solution persistently. Blum and Blesa [6] reset the system when the c_f factor was equal to one.

Venables and Moscardini [45] analyzed the rate of stagnation too, and provided a measurement for the capacitated fixed charge location problem. Let S_{best} be the best solution and $\tau_{i,j}$ be the pheromone of edge (i, j) in S_{best}. Let b be the number of facilities. The stagnation rate is then defined by eq. (3.7). As the ant system converge, the stagnation measurement will approach zero. Once the system stagnated, Venables and Moscardini reset all edge pheromones to τ_{max}.

$$\frac{\sum_{\tau_{i,j}\in S_{\text{best}}} \min\{\tau_{max} - \tau_{i,j}, \tau_{i,j} - \tau_{min}\}}{b} \to 0 \tag{3.7}$$

Liao et al. [29] also reset the pheromone levels when the colony was close to stagnation. To measure the level of stagnation, Liao et al. observed the number of consecutive iterations with a relative solution improvement lower than a specific threshold. When the threshold was reached, both the global solution and the pheromone levels were reset.

It seems apparent that the strategy of resetting the pheromone levels are similar to performing a random restart, and may be perceived not as much a stagnation avoidance strategy as a way to ameliorate the issues with early convergence. It is important to notice that when resetting the pheromone levels, all the knowledge the colony had gained about their environment is expunged. Consequently, ant algorithms that can avoid resetting the pheromone levels may be more efficient.

Minimum Pheromone Threshold Strategy

The Minimum Pheromone Threshold Strategy (MPTS) was proposed by Wong and See [50] in 2009 and produced good results on the quadratic assignment problem [50]. This was an interesting, dynamical approach to pheromone limitation. The lower bound on the pheromone range τ_{min} was adjusted according to the observed convergence rate. When the convergence rate drops, and the ants approach stagnation, the pheromone range is reduced and the ants become more willing to choose edges with low pheromone values.

This was realized with a threshold value $\tau_{thresh} \in (\tau_{min}, \tau_{max})$. At the start of the search, τ_{thresh} is set to some arbitrary value, which is then updated depending on the convergence as the search progress. The value was adjusted by dividing τ_{thresh} by some constant c at fixed intervals. If an ant traverse an edge with pheromone $\tau_{i,j} < \tau_{thresh}$, the edge pheromone $\tau_{i,j}$ would be reset to τ_{max}. Consequently, this strategy effectively prevents the need for reinitializing the pheromone levels. In turn, MPTS also avoid the information loss incurred when resetting the pheromone levels. It appears that the MPTS could be used with any of the aforementioned algorithms, since it does not interfere with the dynamics of the pheromone bounds.

Stagnation in Nature

Ant colonies in nature must explore their environment to discover food in order to avoid starvation. For an ant to be able to scout the surrounding area, the ant must be able to journey outside the established ant trails. If all ants were constrained to only move along the established ant trails, they would become oblivious to the rest of their habitat. This form for stagnation can be observed in nature, and is termed circular milling since these never-ending trails form a ring [40]. Arguably, the stagnation in ACO algorithms can be compared to stagnation in colonies in nature. Balancing exploration and exploitation is apparently important for both realistic and artificial ant colonies.

Ant colonies have been studied in biological research since the late 19th century and empirical studies of insects have provided insight on how insect colonies are capable of avoiding starvation and stagnation [23]. Their success as a complex ecological system has risen from their capability of social organization and labor division [38]. The members of an insect colony need to offer protection, gather food, and attend to the brood. The flexible behavior of the workers is important to maintain the existence of the colony under internal and external challenges [8, 9]. These challenges are overcome by the social organization of individuals. Following the naturalistic solution to stagnation, it would be interesting to incorporate labor division into ant algorithms as a stagnation countermeasure.

3.4 Labor Division

Division of labor can be achieved through worker polymorphism and appear in many termite species and some ant breeds [14, 38, 48]. This section will discuss two labor division models that are observed in nature. First, the Fixed-Threshold model which appear to be a popular abstraction on how a diverse population can exhibit division of labor. Finally, the conceptual Self-Reinforcement model is explored.

Fixed Threshold Model

The fixed threshold model gained concession after Wilson [49] published a seminal paper in 1984 on how one could manipulate different worker castes in the ant genus *Pheidole* to change their behavioral pattern. Wilson artificially removed members from certain castes of a colony and then observed the effect of influencing the ratio between worker groups. The study indicated that members of different castes would eventually carry out the other caste's activities if they perceived that certain tasks were neglected. The analytical success of the threshold model established it as a seemingly popular labor division model in research. Experiments conducted by Robinson [38] indicated that the behavior of the bee family *Apis mellifera* may also be governed by response thresholds using hormonal signals. Furhtermore, Detrain and Pasteels [14] showed that response thresholds govern the actions of the ant species *Pheidole pallidula* in at least two actions: defending their hive and foraging for food. The *Pheidole pallidula* communicate through laying down trails of behavior stimulating different worker classes to join in on specific activities. The stimulus was regulated by the concentration of pheromone on the trail.

Bonabeau et al. [8] publicized a study on how the simple Fixed Threshold Model (FTM) earlier proposed by Robinson [38] can provide the foundation for division of labor in insect societies. The Fixed Threshold model assume the workers in a colony are selectively acting out assignments based on stimuli, where an action is performed if the corresponding stimulus is above its threshold. When the stimulus increased above the threshold, the workers were enticed to carry out the respective activity. Bonabeau et al. [8] focused on how a colony consisting of multiple worker castes with different stimuli thresholds can produce division of labor, and investigated this through quantitative simulations using FTM. They concluded that FTM can account for some of the dynamics observed in nature, such as the plasticity observed by Wilson [49] in the ant genus *Pheidole*. However, FTM is a very simple model and can only account for observations on a small timescale [8]. Nevertheless, FTM appear to be a popular labor division model in literature [5, 8, 14, 22, 26, 35, 36, 44, 48, 49, 51].

The dynamics of the fixed threshold model is governed by two weighted behavior transitions. The probability of starting to perform a certain activity a_i and the probability of stop carrying out activity a_i. The transition between these two states is governed by a stimulus s. An activity is more likely to be carried out, if the corresponding task stimulus is high. The transition probability is regulated by a threshold value θ. Let the probability of starting to perform task a_i be:

$$P_1(a_i = 0 \rightarrow a_i = 1) = \frac{s^2}{s^2 + \theta^2} \tag{3.8}$$

Equation (3.8) describes that an ant may start performing activity a_i with probability P_1 at each time step. After each time step in the model, the agent may cease to perform task a_i with a probability p_{term}:

$$P_0(a_i = 1 \rightarrow a_i = 0) = p_{term} \tag{3.9}$$

Equation (3.9) says that after each iteration, the agent may stop performing a_i regardless of the stimulus level. When activity a_i is executed, the corresponding stimulus s tend to decline. However, the stimulus may still be strong enough to transition the agent to perform the same activity in consecutive iterations, following eq. (3.8).

Arguably, the *Minimum Pheromone Threshold* stagnation prevention strategy described by Wong and See [50], to dynamically adjust the probability of exploring edges with low pheromone strength based on the convergence rate, share some resemblance with a threshold model. In their ACO algorithm, the threshold is represented as the τ_{thresh} criteria in Wong and See [50]. Once the threshold is reached, the corresponding edge is delegated a quantity of pheromone. The additional pheromone would then entice the workers to explore this path.

The only modifications to ACO required in order to incorporate the Fixed-Threshold Model, is to model a stimulus signal. This stimulus could for instance be a stagnation measurement. Thus, a threshold behavior could be executed when the stimulus and stagnation level increase. A fitting behavior would then be to force the ants to explore the landscape.

Self-Reinforcement Model

Another take on labor division is to encourage workers to perform an activity if the work yields positive results. The Self-Reinforcement model strives to represent division of labor as an emergent property from experience, in which the successful execution of an activity should increase the likelihood that the agent will perform the task again. Likewise, should failing the activity lead to a decrease in the probability of performing the activity again. This reinforcement concept has been suggested as governing the control of nest climate in *Bombus terrestris* [47].

It has been shown that this model can promote specialists from a population of initially similar individuals [5, 9]. Theraulaz et al. [44] derived the same conclusion by adding learning to the Fixed Threshold model defined by [7, 8]. Theraulaz et al. extended the FTM by giving each agent an individual threshold level. The threshold would increase or decrease depending on how frequently the agent performed the corresponding activity. From the initial state, where all agents had the same threshold value, Theraulaz et al. achieved a diverse population with workers specialized at different activities due to the variance in their activity stimulus threshold.

3.5 Related Work

Yan et al. [53] were the first to apply ACO to the single-source uncapacitated minimum cost flow problem with concave cost functions. The cost functions they used were square root functions. They created a hybrid ACO algorithm based on Ant Colony System [17], genetic algorithms and threshold accepting. Their algorithm also incorporated a local search. In their tests, they compared the hybrid ACO algorithm to a genetic algorithm and four different local search-based algorithms. The hybrid ACO algorithm and genetic algorithm achieved much better results than the local search algorithms, but on the other hand were much slower. Compared to each other, the hybrid ACO algorithm performed slightly better than the genetic algorithm, both in terms of results and speed.

Monteiro et al. [32, 33] also applied ACO to the single-source uncapacitated minimum cost flow problem with concave cost functions. They looked into three different concave cost functions, among them the fixed charge function. Their ACO algorithm was based on Max-Min Ant System [42, 43], but was adapted to the minimum cost flow problem. One of the alterations they performed was to incorporate a local search, which improved the performance of the algorithm in their tests. The ACO algorithm was compared to a hybrid genetic algorithm and yielded good results and proved to be considerably faster than the genetic algorithm.

In addition to the concave minimum-cost flow problem, ACO has been applied to some similar NP-hard flow and graph problems. Chen and Ting [12] developed two ACO algorithms based on Ant Colony System [17], where one was combined with a lagrangian heuristic, in order to solve the single source capacitated facility location problem. The algorithm combining ACO and the lagrangian heuristic clearly outperformed the other ACO algorithm. Santos et al. [39] developed an ACO algorithm for the capacitated arc routing problem and compared their algorithm to five other metaheuristics. Their algorithm performed well, finding the best known solution to 95% of the test problems.

3.6 Structured Literature Review

The attention to how swarm intelligence and ants in nature could be used to solve optimization problems was sparked by an article by Garnier et al. [23] published in the journal *Swarm Intelligence*. This paper also the triggered the investigation into whether labor division models could be applied to ant algorithms.

To get acquainted with the field of ACO, the seminal book *Ant Colony Optimization* written by Dorigo and Stützle [19] was read. The exploration of this field of study fanned out from this book. *Ant Colony Optimization* listed numerous high quality references.

The second principal paper to this thesis was acquired by skimming through the GECCO proceedings relating to ant algorithms from the previous five years. The GECCO article written by Monteiro et al. [32] concerned how an ant algorithm efficiently solved minimum cost flow problems with concave cost functions (MCFP), which is a very applicable optimization problem. This paper also referenced a public test set with concave cost graph problems created by Beasley [4]. Unfortunately, they did not publish the exact solution results they achieved.

Google Scholar search was used to investigate the MCFP domain and retrieve relevant papers. The search keyword sentences was *concave cost graph problems* and *transportation problems*. The search retrieved an abundance of papers that were filtered out based on their relevancy and citation count. The refined search returned seven central papers [1, 24, 27, 37, 46, 54, 55].

Google Scholar's *related articles* functionality was used to retrieve relevant articles on popular labor division models in literature.

3.7 Summary

This chapter has discussed the four principal themes of this report. First, the reasoning for why and additional test set of small-world graphs were generated and why these graphs provide a more realistic benchmark test for the transshipment problem. Second, the chapter reviewed important aspects of concave cost functions in flow problems. The most essential characteristic was that the optimal solution in an MCFP is always an arborescent route. Third, the chapter surveyed the various stagnation avoidance strategies used in literature. However, no research was discovered on how to apply biological labor division models to ACO. Finally, the chapter reviewed the two labor division models: Fixed Threshold and Self-Reinforcement model. The two models were defined at different levels of formality. While the Self-Reinforcement model is a concept, the Fixed Threshold model is defined by two equations.

The next chapter will first examine how an ant should construct a solution in the MCFP domain, and then move on to specifying how the two nominated labor division algorithms was incorporated into ACO.

Chapter 4

Methodology

4.1 Solution Construction

4.1.1 Introduction

The minimum cost flow problem can intuitively be represented as a graph problem where the goal is to find the cheapest way to flow some commodity through a network. This section will discuss two solution construction methods proposed to generate ant solutions in the MCFP domain. First, a naturalistic approach to solving MCFP is formulated. The second strategy is an enhanced extension the first approach, where the solution construction is oriented toward computational efficiency.

4.1.2 Alternated Move and Push

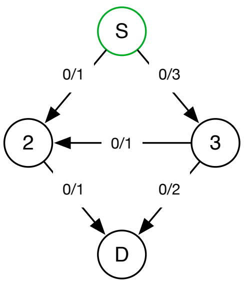
This first approach is also the most naturalistic solution construction method that was considered. The idea was to repeatedly let an ant walk along a path from the source node to a sink node until the ant trails formed a network that connected all demand nodes to the source node. Empirically, this approach was strongly rooted in nature, where ant colonies have to organize trail systems leading to food sources in their environment.

This strategy entails an incremental solution construction, where a single solution is constructed by letting the ant walk through the graph multiple times. An illustration of the procedure is depicted in Figure 4.1. At each junction faced by the ant, it chooses one of the radiating paths randomly. Expressed in a different way, the solution is constructed by a series of local decisions and can be characterized as a decentralized strategy.

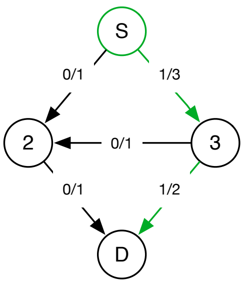
The *Alternated Move and Push* strategy show how a system of simple agents are able to cooperate through interactions with the environment to solve flow path problems. This strategy is a general solver for capacitated edge problems. Whether the problem is to find the minimum flow cost, or to find the shortest TSP path; as long as the problem can be described as a graph, this procedure be used as a meta-solver.

In order to use this approach to solve problems in the MCFP domain, we have to transform the uncapacitated problem into a capacitated graph with a single source. This is easily achieved through a three step process. First, convert the graph to a directed graph and add a super-sink. The super-sink will be the only demand node. Second, add edges leading for each of the previous demand nodes into the super-sink. Finally, assign infinite capacity to the internal edges, while the edges leading to the super-sink has capacity equal to the demand of the originating node. The result of this process is depicted in Figure 4.2. The demand from demand nodes $D = \{2, 3, 4, 5\}$ has been replaced by capacitated edges leading to the super-sink. The capacity of the corresponding edge mirrors the demand of the emitting node.

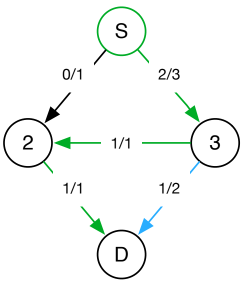
Unfortunately, this solution construction procedure can also create solutions that are guaranteed not to be optimal solutions in MCFP. This issue arises since the edges are undirected in the MCFP domain, but need to be directed in *Alternated Move and Push* strategy. The result is that ants may create naive paths, such as the example illustrated in fig. 4.3. The figure shows how the *Alternated*



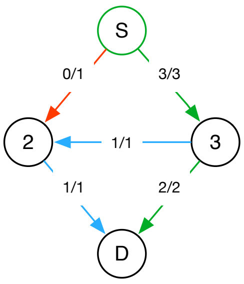
(A) A capacitated graph with one source S and one sink D node.



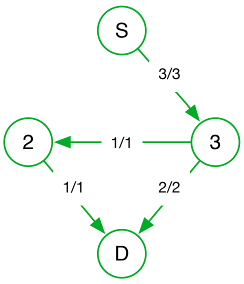
(B) The first trail created by an ant. The path is randomly created based on pheromone levels. A single flow unit is pushed along the the edges included in the trail. The trail is denoted by a green line.



(C) A second pass is conducted by the same ant. This resulted in a new random path. A single flow unit is pushed along the green trail. The blue line denotes an edge that has been previously traversed by the ant.



(D) A third pass is performed by the same ant, and a flow unit is pushed through the graph along the green path. The red line denotes an edge that cannot be traversed since the node it connects to has not outgoing capacity. The red edges will be removed from the solution.



(E) The ant can no longer reach the sink node due to the edge capacities. The ant therefore terminates its search for paths through the graph. The solution construction has finished.

FIGURE 4.1: How a solution is constructed using the Alternated Move and Push strategy.

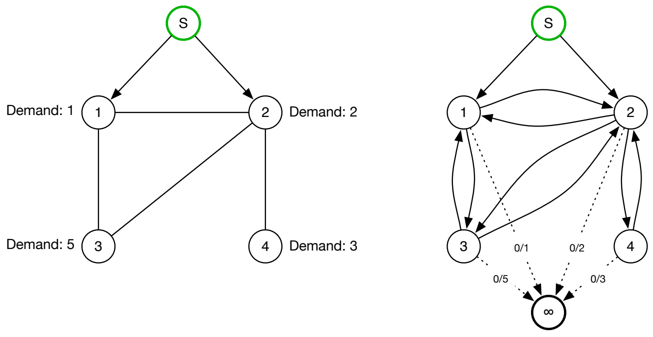
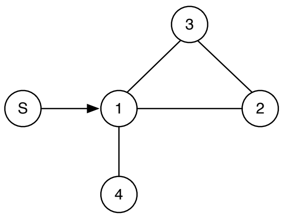
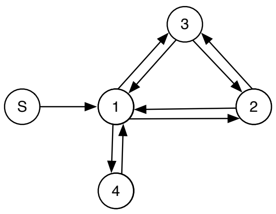


FIGURE 4.2: This figure show how a undirected single source flow network with multiple demand nodes and uncapacited edges must be converted to a directed graph with a single sink before applying the *Alternated Move and Push* strategy.

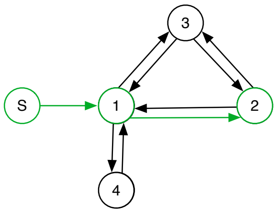
Move and Push procedure could end up constructing circular flow routes. In worst-case situations, the problem showcased in fig. 4.3 could prevent the search from ever finding the optimal solution.



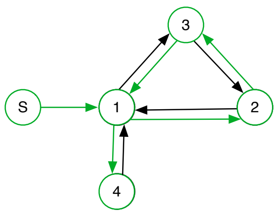
(A) An undirected graph where each edge has the same cost.



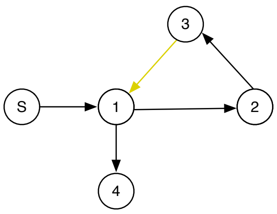
(B) The graph must be converted to a directed graph before the *Alternated Move and Push* strategy can be used to construct a solution.



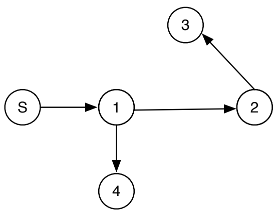
(C) One ant has begun its tour of the graph. The green lines denote the edges that have been traversed. The randomly created path is $S, 1, 2$. The ant is position in node 2. From this state, the rest of the path is randomly chosen to be $3, 2, 4$.



(D) The ant has completed its tour of the graph. The green edges indicate the constructed route.



(E) The solution constructed by the ant has clearly included an unnecessary edge. Remember that the optimal solution to a MCFP is an arborescent path. Thus, the optimal solution would not include the yellow edge.



(F) This would have been the ideal route to construct in an uncapacitated minimum cost flow problem.

FIGURE 4.3: This is an illustration of how the Alternated Move and Push strategy can construct solutions that are guaranteed to be suboptimal in a MCFP with concave cost functions.

4.1.3 Randomized Spanning Tree

When examining figs. 4.1 and 4.3, it is apparent that the constructed solutions share similarities with a spanning tree generation. The ant is consecutively faced with edge conjunctions in the graph until it reaches the super-sink. Upon reaching the super-sink, the ant will have covered the demand of one demand node. Then the ant is transported to the source node and will generate a path to another demand node. This procedure can be efficiently implemented as a randomized spanning tree (RST) construction.

The spanning tree is incrementally constructed by selecting edges based on the standard pheromone selection scheme as described in section 2.3. Each spanning tree start off with the source node as the only vertex in the tree. In the first step of the tree construction, one of the edges emitting out of the source node is randomly chosen. The nodes that are connected to the source nodes will then be appended to a frontier set. During the subsequent construction steps, any edge emitting out of the frontier set may be selected to extend the spanning tree. This process is repeated until all demand nodes are reachable from the source node. An illustration depicting this process is presented in fig. 4.4.

Arguably, the solution construction would still be a series of local decisions on the solution constructing level and constitute a decentralized solution construction mechanism.

Contrary to the *Alternated Move and Push* strategy, the RST solution construction is capable of working with both directed and undirected graphs. Therefore, RST can be applied to MCFP without preprocessing the graph. The RST construction procedure would only require the ant to "walk" through the graph once rather than the repeated instigation required in Move and Push. The ant only need to traverse the graph once, since the edge flow would be entailed by the set of selected edges.

An important advantage of using the RST approach is that the ants will avoid the problem discussed in the Move and Push strategy where the optimal flow

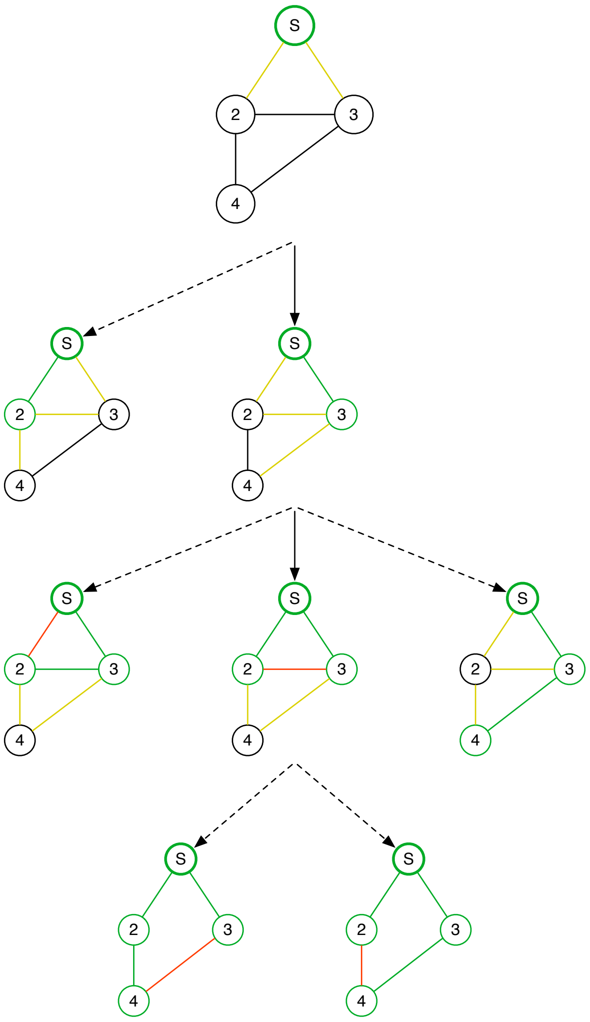


Figure 4.4: Constructing a solution with the randomized spanning tree procedure. The solid black arrow denote the resultant path of constructing the partial solutions. In the initial state of the search, the ant can only select edges emitting from the source node. In all subsequent states, the ant can traverse any node emanating from the frontier set of green nodes. The selectable edges in each state are colored yellow. The procedure select one of the yellow edges according to the calculated ACO edge transition probabilities as specified in eq. (2.5). Illegal edges, pointing to nodes within the frontier set, are marked red and removed from the solution. The construction terminates when there are no more valid edges to traverse.

path might not be obtainable by the ants. In addition, the RST method will only construct arborescent flow routes, which are known to be the optimal solution candidates in MCFP.

It is worth noting that the RST construction is biased because edges that are added to the frontier set early in the construction procedure are more likely to be chosen than edges that are added to the frontier set later on. One of the effects of this bias, is that a uniform pheromone distribution does not entail a uniform probability distribution over the solutions space. This is illustrated in fig. 4.5.

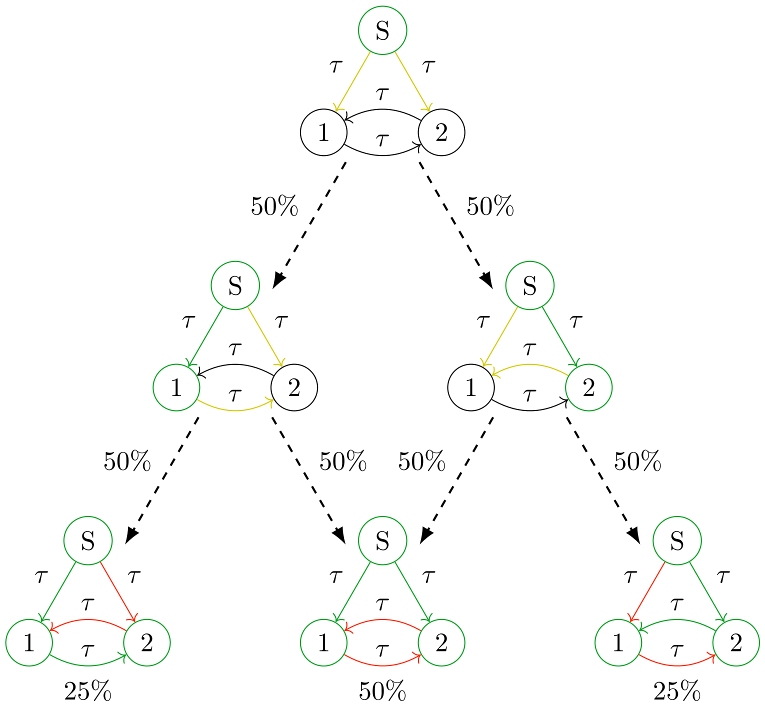


FIGURE 4.5: An example illustrating that a uniform pheromone distribution does not necessarily define a uniform probability distribution over the solution space. Every edge has an equal amount of pheromone τ. Yellow edges indicate edges that may be chosen to expand the arborescence, while green edges are included in the current tree. Red edges cannot be chosen because doing so would break the arborescence.

4.2 Self-Reinforcement ACO Algorithm

4.2.1 Introduction

Self-Reinforcement ACO (SR-ACO) is the first novel combination of an ant algorithm and a labor division model proposed in this paper.

4.2.2 Implementation

The main idea behind Self-Reinforcement ACO is to only reinforce a solution once. This ensure that the colony is focused toward exploration since each solution is only reinforce once, while at the same time it ensures exploitation since it is only reinforcing the best possible solution. This was achieved by both maintaining a set of explored solutions and a priority queue of unexplored solutions. After a solution has been constructed the algorithm checks whether the solution is in the set of explored or unexplored solutions. If the newly constructed solution is not in either of the sets it is added to the priority queue of unexplored solutions, otherwise it is discarded.

During pheromone updating, the head of the priority queue of unexplored solutions is retrieved, reinforced and finally added to the explored set. This way of choosing which solution to reinforce makes SR-ACO a very exhaustive search. Provided enough iterations, SR-ACO will reinforce all discovered solutions and thus search a large portion of the search space. Due to its probabilistic nature it is not guaranteed to discover all the solutions and thus find the optimal solution.

Because a solution will only be reinforced once during a search, SR-ACO require a large number of ants and a high pheromone evaporation rate to ensure that the solution is exploited to a large enough extent before the subsequent solution is tried. A problem with high evaporation rate is that an edge may quickly lose nearly all of its pheromone and become almost impossible to select during solution construction. In order to avoid this problem, SR-ACO utilizes pheromone bounds to ensure that no edge is impossible to choose during solution construction.

In Self-Reinforcement ACO the pheromone bounds are explicitly defined. Given a solution S to reinforce, the pheromones are updated according to:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}(s) \qquad \forall (i,j) \in E \tag{4.1}$$

where

$$\Delta\tau_{ij} = \begin{cases} \tau_{\min} & \text{if edge } (i,j) \text{ is not in the solution } S \\ \tau_{\max} & \text{if edge } (i,j) \text{ is in the solution } S \end{cases} \tag{4.2}$$

As long as the initial pheromone level τ_0 is not greater than τ_{max} or less than τ_{min}, this updating function ensure that the pheromones are always between τ_{min} and τ_{max}.

The pheromone bounds both regulate the degree of stigmergy and how similar the generated solutions will be to the reinforced solution. In this context, similarity is measured as the number of edges common to the two solutions. The closer the ratio $\frac{\tau_{\min}}{\tau_{\max}}$ is to one, the less stigmergy and similarity between the reinforced solution and the generated solutions there will be. Likewise, the closer the ratio is to zero the more stigmergy and similarity between the reinforced solution and the generated solutions there will be.

Due to the high evaporation rate and that only the best solution in the priority queue is reinforced, it is expected that SR-ACO will have a fast convergence in the beginning of the search. Then, when SR-ACO reach a local optimum it will successively try increasingly worse solutions in an effort to escape the local optima. Consequently, the average cost of the generated solutions is expected to increase until an improvement upon the best discovered solution is found.

SR-ACO is designed to be very resistant to stagnation as it always reinforces new solutions and the pheromone bounds should ensure that the search always explores new areas of the search space.

One drawback of maintaining the set of explored solutions and the priority queue of unexplored solutions is the increased memory usage. However, it is possible to limit the memory usage by setting a fixed capacity for the priority queue. In the experiments carried out in this study, the capacity for the priority queue was set equal to the number of iterations. The set of explored solutions is implicitly limited to the same number of elements, since exactly one solution is added to it every iteration.

4.2.3 Search Dynamics

Figure 4.6 shows an illustration of how the priority queue of unexplored solutions affects the search. The blue circle indicates the best solution at each iteration, which is then removed from the queue and reinforced. New solutions inserted into the queue in the subsequent iteration is pointed to by dashed arrows. Red circles indicate solutions that are removed from the queue.

The figure depict how the priority queue enable SR-ACO to concurrently optimize multiple facets of the best solutions, by selectively focusing on only the currently best solution in the priority queue. In the initial phase, the search focuses on finding the best solutions in the neighborhood of the reinforced solution. This behavior continues until the search encounters a local minimum. If the search has reinforced a locally optimal solution, it will defer optimizing this solution sequence and rather reinforce the second best solution in the subsequent iteration.

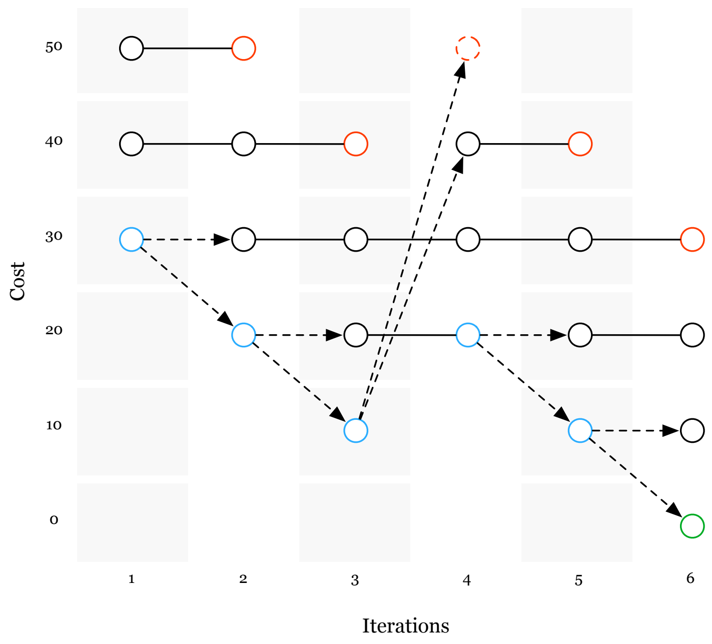


Figure 4.6: Illustration of the search progression in SR-ACO. The circles represent solutions maintained in the SR-ACO priority queue. The priority queue contains the three solutions with the lowest cost. At each iteration, one solution is removed from the queue and two new solutions are generated by the ants. The dashed lines indicate that the targeted circle was generated from the previous solution. The solid lines indicate that the solution from the previous iteration was retained in the priority queue. The red circles have been replaced in the priority queue by better solutions. In iteration three, we observe that the search defers optimizing the previously best solution and continue optimizing the second best solution. This behavior occurs when the search enters a local optimum.

4.3 Fixed-Threshold ACO Algorithm

4.3.1 Introduction

The Fixed-Threshold ACO (FT-ACO) algorithm is the second algorithm proposed in this article. FT-ACO is a combination of the Fixed-Threshold labor division model and an ant algorithm. The intention of incorporating the threshold labor model was to trigger the ant colony toward exploring the search landscape whenever the search approach stagnation behavior.

4.3.2 Implementation

FT-ACO is constituted by two components. First, the ACO algorithm responsible for discovering the solution. Finally, the Fixed-Threshold system that trigger scouting. In addition, the Fixed-Threshold model also requires an arbitrary function that evaluate whether the system is closing in on stagnation.

The ant algorithm in FT-ACO is similar to Max-Min Ant System (MMAS), except that the pheromone bounds have been redefined. The upper pheromone boundary τ_{max} is removed and the lower boundary τ_{min_i} is defined per node i. The lower boundary is defined as 0.05 multiplied by the highest pheromone level among the incident edges. In other words, if there are only two edges out of a node then the edge with the lowest pheromone level will have at least a five percent probability of being selected. Let S be the set of edges connected to node i.

$$\begin{aligned}\tau_{max} &= \infty \\ \tau_{min_i} &= 0.05 \max_{ij \in S} \tau_{ij}\end{aligned}$$

FT-ACO only reinforces the edges present in best found solution. The same strategy is also used in MMAS.

The rest of this section will explain the inner workings of how the Fixed-

Threshold model has been incorporated and how stagnation is used as a stimulus. An extensive analysis on the decision of the arbitrary evaluation function is presented in section 5.3.5.

4.3.3 The Fixed-Threshold Model and Scouting

The Fixed-Threshold model was incorporated into ACO to provide a black box mechanism for balancing the focus between exploration and exploitation. The Fixed-Threshold ACO algorithm became a process that alternates between focusing on exploration and exploitation.

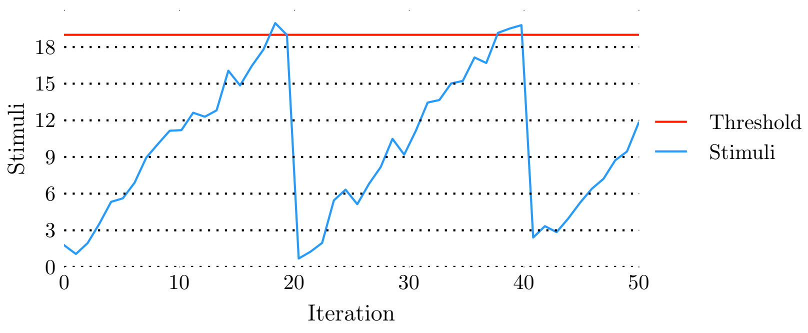


Figure 4.7: A range bound stimulus variable. When the stimulus signal has accumulated enough strength, the fixed threshold is intersected and the associated behavior may be triggered. The behavior counters the stimulation, and a sudden drop is observed in the stimulus. In FT-ACO, the stimulus is produced by stagnation and the triggered behavior is the scout reaction.

Figure 4.7 illustrate how the threshold behavior will be triggered based on a stimulus. When the stimulating signal accumulated to the fixed threshold level, a specific behavior was triggered. In this case, the threshold dictates whether the ant colony should focus on exploration or exploitation. The stimulus was generated by an arbitrary stagnation evaluation function.

Let Distance() denote the arbitrary stagnation measuring function. This function takes the best found solution S_{best} and the best solution S_k in iteration k as parameters. The stimulus s_k at iteration k was calculated as:

$$s^k = \begin{cases} s^{k-1} + 1 & \text{if Distance}(S_{\text{best}}, S_k) = 0 \\ \frac{s^{k-1}}{\text{Distance}(S_{\text{best}}, S_k)} & \text{otherwise} \end{cases} \tag{4.3}$$

When the stimulating signal is weak, the ants will focus on exploitation and prioritize to traverse edges with higher pheromone levels. This phase corresponds to the area below the red threshold line in fig. 4.7. As the stimulus cross the red line in fig. 4.7, the colony will shift the focus to scouting the graph. The focus is shifted by smoothing the edge transition probabilities in the ACO algorithm. Let S be the set of edges currently visible to an ant, τ_{ij} denote the pheromone level and η_{ij} denote the visibility on the edge spanning nodes i and j. Furthermore, let α and β be parameters that govern the relative importance of τ and η. The standard transition probability P_{ij} of an edge is defined as:

$$P_{(i,j)} = \frac{(\tau_{(i,j)})^{\alpha}(\eta_{(i,j)})^{\beta}}{\sum\limits_{(i,j)\in S} (\tau_{(i,j)})^{\alpha}(\eta_{(i,j)})^{\beta}} \tag{4.4}$$

The transition probabilities were smoothed by adding a quantity of pheromone inversely proportional to the difference between the strongest and weakest pheromone signal an ant perceived. In other words, edges with lower pheromone level would be compensated with more pheromone.

An exploration factor γ was also introduced. The exploration factor specified the extent of how even the probabilities should be turned. The parameter γ take on values in the range $[0, 1]$, where zero imply no regularization and one produce completely homogenized transition probabilities. Let S be the set of edges currently visible to an ant. The smoothed probability P^*_{ij} of edge ij is defined as:

$$P^*_{(i,j)} = \sum_{(i,j)\in S} P_{(i,j)} + \gamma \left(\max_{(u,v)\in S} P_{u,v} - P_{(i,j)} \right) \tag{4.5}$$

The effect of the smoothing function is illustrated in fig. 4.8. The figure clarifies

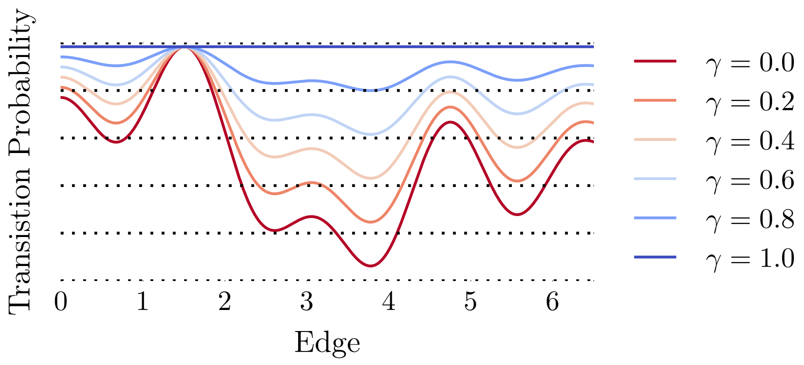


FIGURE 4.8: Illustrating how the smoothing function defined in eq. (4.5) manipulate transition probabilities. As the probability distribution become plane, the transition probabilities become uniform. Consequently, the scouts will become indifferent to the differences in pheromone signal. This is beneficial when exploring the search landscape, but may prevent scouts from discovering low cost solutions.

how the exploration factor γ can be used to adjust how exploratory the scouts behave, through smoothing out the transition probabilities. A γ value equal to zero, will render the scouts oblivious to the actual differences in the transition probabilities, while a γ value equal to one will make the scouts very likely to construct a path equal to the best know solution. The γ value should therefore be defined as some intermediate value, typically closer to one than zero in order to encourage scouts to explore routes in the proximity of the best discovered solution.

The relationship between γ and expansion of the search area is tentatively illustrated in fig. 4.9. The colored regions symbolize the probable nodes a scout may reach. The set of reachable nodes expand increasingly fast by each node the ant traverse. The search region expands because the scouts modify their its transition probabilities such that edges with little pheromone are more likely to be traversed. During the initial moves in the graph, the scout is still likely to wander along a path with high pheromone levels, but the odds of deviating from

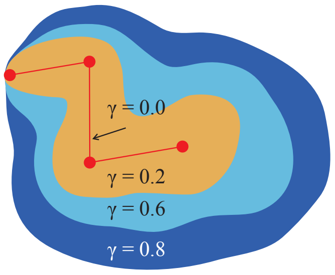


Figure 4.9: Illustrating how γ in eq. (4.5) can be adjusted to expand the regions of the search space reachable for a scout. If the exploration factor γ is close to zero, the scouts will be more likely to explore regions of the graph closer to the best solution. Contrarily, an exploration factor γ equal to one would allow the scouts to explore the entire landscape.

that path increase by each junction.

It is clear that the smoothing performed by eq. (4.5) provide FT-ACO with the means to manipulate the focus on exploration in the vicinity of the best solution according to the parameter γ.

The decision on when to shift the predominance of the focus from exploration to exploration is governed by the Fixed Threshold Model. Let the stimulus s denote the measured degree of stagnation and θ be the threshold limit. Then, let P_{explore} be the probability of scouting during the next iteration. Following the work of Bonabeau et al. [7–9] and Theraulaz et al. [44], the threshold function employed in FT-ACO was identical to these papers and defined as:

$$P_{\text{explore}} = \frac{s^2}{s^2 + \theta^2} \tag{4.6}$$

Equation (4.6) specify the probability for the colony to start exploring based on a stimulus s and a threshold θ in the subsequent iteration. In other words, when the colony is triggered to explore, it will do so for one iteration. Following an exploratory iteration, the colony will fall back to exploitation if the stimulus has dropped sufficiently. If the stimulus did not drop enough, the colony will still have a high probability of executing another exploratory step.

During the exploration phase, the colony is expected to discover poor solutions.

The poor scout solutions will then be optimized during the succeeding exploitation phase. This idea was realized through a three step process. First, designate the best solution constructed by the scouts in the exploration phase as the reinforced solution. Second, reset the pheromone on all edges to $0.05\tau_0$. Finally, deposit some pheromone τ_0 on the edges in the reinforced solution. This will effectively make the scouted solution become a breadcrumb trail through the graph. The trail will encourage the foragers to look for solutions in the neighborhood of the newly explored trail.

4.3.4 Using Stagnation as Stimulus

The aim of incorporating labor division into ACO was to avoid stagnation. An ACO system is said to enter stagnation when all the ants in the colony produce the same solution. In other words, stagnation occur when the colony fail to discover new solutions. It therefore seems fitting to use stagnation as the stimulating signal.

In order to use stagnation as the stimulus we need to measure how close to stagnation the colony is. A review on how the stagnation measurement function was selection is presented in section 5.3.5. Through experimentation the stagnation measurement was decided to be a comparison of the best found solution S_{best} and the iteration best solution S_k. An efficient approach to compare solutions the MCFP domain is to use the Euclidean norm.

The stimulus s should decrease proportionally to the difference between S_{best} and S_k such that the threshold behavior can be postponed if the ant colony manages to avoid stagnation without scouting. When the iteration solution is similar to the global solution, the search tends to be closer to stagnation than if the iteration solution is very different. This can be achieved by defining s as:

$$s^k = \begin{cases} s^{k-1} \frac{1}{\|S_{\text{best}} - S_k\|} & \text{if } \|S_{\text{best}} - S_k\| > 0 \\ s^{k-1} + 1 & \text{otherwise} \end{cases} \tag{4.7}$$

where $\|S_{\text{best}} - S_k\|$ is the Euclidean norm between the two solutions.

Equation (4.7) dictate that the stimulus should increase whenever the global and iteration solutions are equal. However, if the solutions are different the stimulus s will decrease inversely proportional to the Euclidean distance between S_k and S_{best}.

4.3.5 Dynamic Deposit Quantity

When the ants have constructed their solutions, a portion of pheromone is deposited on the edges in the best solution S_{best}. In turn, this will make the edges pertaining to the best solution to be more likely to traverse in the subsequent iteration as well. The pheromone levels are not upper bounded in FT-ACO, like it is in MMAS. By removing the upper bound τ_{max}, the algorithm will converge faster. A fast convergence is preferable in FT-ACO, since this will in turn increase the stimulus for scouting the environment.

The quantity of pheromones deposited $\Delta\tau_{ij}$ is in the range:

$$\Delta\tau_{ij} \in [0, \rightarrow\rangle \tag{4.8}$$

Let C_k denote the cost of the best found solution S_{best} until iteration k. Then $\Delta\tau_{ij}$ is defined as:

$$\Delta\tau_{ij} = \begin{cases} \frac{C_{k-1}}{C_k}, & \text{if } (i,j) \in \text{best tour.} \\ 0, & \text{otherwise.} \end{cases} \tag{4.9}$$

The expression $\frac{C_{k-1}}{C_k}$ try to capture the relative improvement of newly discovered solutions. The quantity of pheromones deposited is proportional to the improvement of the solution cost. This will inherently make an effort to guide the ACO search toward promising portions of the search landscape. In addition, this pheromone strategy will influence the ant colony to explore the neighborhood of the newly discovered solution rather than optimizing a previously best solution.

The shift in focus is rooted in the potentially large quantities of pheromone that can be deposited by this strategy.

An alternate approach to influence the colony to focus on new, improved solutions may be to increase the evaporation rate after an iteration where an improved solution is discovered. This is better illustrated by an example. Lets say the graph has three edges with two pheromone units each, and that we have discovered a relative solution improvement of 3.0. The *Dynamic Deposit Quantity* strategy dictate that the edges present in the best solution should be reinforced with three pheromone units. Arguably, the same shift in prioritization could be achieved by evaporating 60% of the previous pheromones and only deposit one pheromone unit.

The intuition of guiding the search toward regions of the search space, where newly discovered solution improvements were found, seems reasonable. The result of this modification is discussed in section 5.3.9.

4.4 Experimental Plan

4.4.1 Introduction

This section will go over how the experiments were conducted, and provide reasoning for the algorithms we decided to include in the experiment. Finally, the hand-tuned parameters used by the algorithms are presented.

4.4.2 Outline

The purpose of the experiments was to evaluate the performance and stagnation avoidance ability of FT-ACO and SR-ACO. In order to achieve this, the two algorithms were compared with two other ACO algorithms, Max-Min Ant System and Rank Based Ant System, on two problem sets consisting of 300 flow networks in total. Because ACO algorithms are probabilistic and can produce different results each time, the algorithms were run 100 times on every flow problem in order to provide accurate statistical data.

Max-Min Ant System was chosen for comparison because it has previously been applied to concave minimum-cost flow problems by Monteiro et al. [32, 33] and includes a stagnation avoidance mechanism. It is also one of the most studied ACO algorithms [19, p. 76]. On the other hand, Rank Based Ant System was chosen because it is a simple ACO algorithm without any stagnation avoidance mechanism [11]. Due to the lack of a stagnation avoidance mechanism, the results of AS$_{rank}$ could be used as a point of reference on how important stagnation avoidance is in a given flow network. The four algorithms used the same solution construction method and thus only differed in how the pheromones were updated.

4.4.3 Experiment A: Convergence

The first experiment that was carried out, evaluated whether the proposed incorporation of labor division either positively or negatively affected the convergence of ACO. The experiment focused on solving the problem instances of both the random graph set from Beasley [4] and the small-world graph set. These runs could then provide a statistical basis for analysis.

4.4.4 Experiment B: Stagnation Avoidance

The second experiment focused on the stagnation avoidance capability of MMAS, AS$_{rank}$, SR-ACO and FT-ACO. The result of AS$_{rank}$ would then be used as a point of reference to how important stagnation avoidance is in a given problem instance. Similar to experiment A, the experiment consisted of solving the problem instances of both the random graph set from Beasley [4] and the small-world graph set 100 times.

4.4.5 Problem Sets

The four algorithms were run on two problem sets. The first problem set was provided by Beasley [4] and was also used by Monteiro et al. [32, 33] to evaluate their ACO algorithm. It contained 240 flow networks, ranging in size from 10 to 50 nodes. The flow networks were divided into groups of varying ratio between unit costs and start-up costs with three networks in each group. An overview of the groups and their respective ratios can be seen in table 4.1.

TABLE 4.1: An overview of the groups and the respective ratio of start-up costs to unit costs in the graph set provided by Beasley [4]. For the largest networks with 40 and 50 nodes, only the five first groups were present. A ratio of ten denote that the unit cost was one tenth of the start-up cost on the edge.

Group	**Ratio**
1	300
2	30
3	10
4	3
5	1.5
6	550
7	55
8	5.5
9	2.75
10	0.55

In addition, we generated a second set of 60 scale-free small-world networks using the algorithm from Holme and Kim [25]. A scale-free network is a network in which the degree distribution follows a power law, while a small-world network is a network in which there on average is a short distance between all pairs of nodes. These properties were chosen because they appear in many real networks, including transportation networks. Since the scale-free and small-world properties are more apparent in larger networks, the generated networks range in size from 40 to 100 nodes. Like the flow networks provided by Beasley [4], the scale-free small-world networks were also divided into groups with varying ratio between unit costs and start-up costs. An overview of the groups and their respective ratios can be seen in table 4.2.

TABLE 4.2: An overview of the groups and the respective ratio of start-up costs to unit costs in the small-world graph set generated by us. A ratio of two denote that the unit cost was half of the start-up cost on the edge.

Group	Ratio
1	100
2	2
3	1
4	0.67
5	0.5

4.4.6 Measuring Stagnation

One of the central focuses of the experiments was to investigate the ability of FT-ACO and SR-ACO to avoid stagnation. Therefore, after running the algorithms, a separate routine was executed to detect whether the search had stagnated. The stagnation detection routine is listed in algorithm 4.1. Let K be the number of iterations and M be the number of ants. Furthermore, let C_k^m be the cost of the solution constructed by ant m in iteration k. The routine assigned the best solution in each iteration C_k. A successful convergence by an algorithm could then be identified by observing the decrease in C_k. If the decrease in C_k halted before reaching the global minimum C_{OPT}, and $C_K = C_{K-1}$, the search had stagnated.

Algorithm 4.1: Stagnation detection

```
Result: $true$ if avoided stagnation, otherwise $false$
for $k \leftarrow 1$ to $K$ do
    $C_k \leftarrow \min_{m \in [1,M]} C_k^m$
return $(C_K = C_{\text{OPT}}) \vee (C_K > C_{K-1})$
```

4.4.7 Parameter Values

In order to ensure a fair comparison between the algorithms, the algorithms were run with optimized parameter configuration. The optimized values were found by testing multiple configurations of values on all flow networks from the problem sets and selecting the best performing configuration for each algorithm. The only requirement for the parameter values was that the number of generated solutions, which is the product of the number of iterations and the number of ants, had to be equal to 24000. As long as this requirement was met, the algorithms could either favor using a lot of ants or running for many time steps. The parameter values used by each algorithm during the final experiments are listed in table 4.3.

TABLE 4.3: This table list the specific parameter configuration obtained after the extensive parameter optimization runs.

Algorithm	**Parameter**	**Value**
AS_{rank}	I	600
	M	40
	ρ	0.1
	α	1.0
	β	1.0
MMAS	I	600
	M	40
	ρ	0.01
	α	1.0
	β	1.0
SR-ACO	I	100
	M	240
	ρ	0.8
	α	1.0
	β	1.5
	τ_0	1.0
	$\tau_{\min}$	0.011
	$\tau_{\max}$	1.0
FT-ACO	I	600
	M	40
	ρ	0.3
	α	1.0
	β	2.0
	γ	3.0
	θ	100

4.5 Summary

In this chapter we first covered how an ant construct a solution. The construction strategy was derived from how simple agents could produce valid solutions to a MCFP. This refinement process resulted in the Randomized Spanning Tree (RST) strategy where a solution is constructed as a spanning tree using the ACO edge transition probability function. An important advantage with RST was that it produced arborescent flow routes, which are known to be the optimal solutions to MCFP.

The chapter has also described how SR-ACO employs the self-reinforcement concept to balance exploration and exploitation, while FT-ACO utilized the Fixed-Threshold Model as a black box mechanism to avoid stagnation. In the next chapter, these techniques will be scrutinized by analyzing their performance on two large problem sets. The examination will help provide answers to the research question posed in chapter one.

Chapter 5

Results and Discussion

5.1 Introduction

This chapter presents and discusses the results from the experiments outlined in section 4.4. The experiments were aimed at investigating how the three ACO algorithms Max-Min Ant System (MMAS), Fixed-Threshold ACO (FT-ACO), and Self-Reinforcement ACO (SR-ACO) perform on minimum cost flow problems and if labor division is a better option for avoiding stagnation than the pheromone limiting scheme used in MMAS. In order to determine the stagnation avoidance performance of the three algorithms, they were compared to AS$_{rank}$, an ant algorithm without any stagnation avoidance mechanism.

The algorithms were run on two sets of flow networks; one set by Beasley [4] consisting of 240 flow networks ranging from 10 to 50 nodes, and one set generated by us of 60 small-world flow networks ranging from 40 to 100 nodes. The results on all flow networks are presented in their entirety in appendices A and B and provide statistical justification for the claims made in this chapter. A summary of the results showing the percentage of flow networks on which the algorithms stagnated, did not converge or found the optimal solution can be seen in table 5.1

and fig. 5.1.

5.2 Results Overview

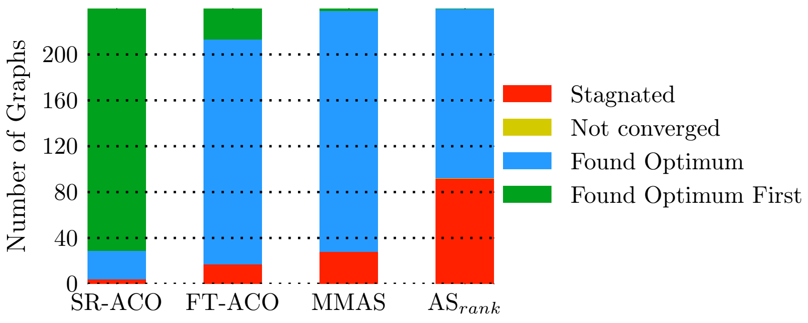
As can be seen in fig. 5.1, all of the stagnation avoiding algorithms performed better than AS$_{rank}$, especially on the small-world networks. On the flow networks by Beasley [4] the three algorithms were able to find the optimal solution in almost equally many problems, but the two labor division algorithms outperformed MMAS on the small-world set.

Table 5.1: A table showing the stagnation occurrences calculated by algorithm 4.1. The row *Not yet converged* indicate that the search had not yet entered stagnation, nor had it found the optimal solution yet. The row *Avg. Sol. required* list the number of solutions that had to be generated on average before the given algorithm found the global optimum. The row *Avg. Sol. required* only consider the 149 problem instances where all of the algorithms found the global optimum.

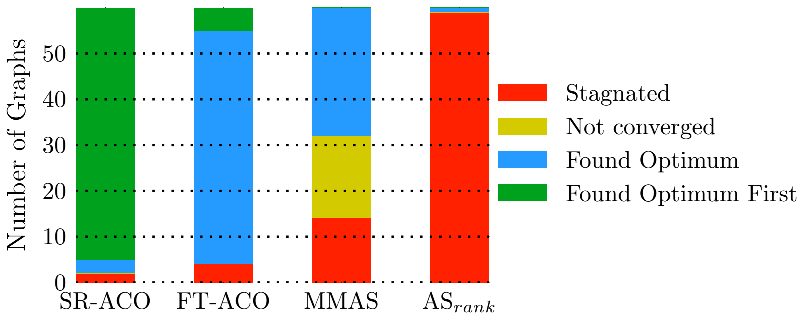
	SR-ACO	**FT-ACO**	**MMAS**	AS$_{rank}$
Found Optima First	88.7%	10.7%	0.7%	0%
Found Optima	98%	93%	80%	49.7%
Stagnated	2%	7%	14%	50.3%
Avg. Sol. required	1045.64	1832.48	5259.33	3606.17

The stagnation measurements produced by algorithm 4.1 as defined in section 4.4.6 clearly indicated a difference in the performance of the ant algorithms. SR-ACO appeared to be the most efficient strategy, discovering the optimal solution first in almost 89% of the test graphs. The two labor division based algorithms, FT-ACO and SR-ACO, both performed very well, respectively solving 93% and 98% of the problem instances. The algorithm used as a stagnation avoidance benchmark, MMAS, found the global optimal solution in 80% of the test cases. This indicates that the labor division models can be efficiently applied to ACO to avoid stagnation, at least in the MCFP domain. The results of algorithm 4.1 are listed in table 5.1 and illustrated in fig. 5.1.

In comparison to AS$_{rank}$, MMAS and the proposed algorithms clearly achieved



(A) A bar chart describing the performance of SR-ACO, FT-ACO, MMAS and AS_{rank} on the graph set created by Beasley [4].



(B) A bar chart describing the performance of SR-ACO, FT-ACO, MMAS and AS_{rank} on the Small World graph set.

Figure 5.1: Two bar charts summarizing the result of the experiment and illustrating the performance of MMAS, AS$_{rank}$, SR-ACO and FT-ACO.

much better results. As depicted in table 5.1, AS$_{rank}$ tend to suffer from early convergence. AS$_{rank}$ performed especially poor on the large flow problems in the small-world graph set. AS$_{rank}$ was only able to solve two percent of these problems, and stagnated in the remaining 98% instances.

5.3 Results Analysis

5.3.1 Search Performance

MMAS has previously been applied to the single source uncapacitated minimum cost flow problem by Monteiro et al. [32]. Their paper stated that MMAS was an efficient solver in this problem domain. Consequently, MMAS was included in the experiment as a benchmark for the proposed algorithms.

As expected, MMAS proved to be less prone to stagnation than AS$_{rank}$, at the cost of having a slower convergence. As a result of the stagnation avoiding mechanism in MMAS, the algorithm discovered the optimal solution in 80% of the flow problems. This is a clear improvement over AS$_{rank}$, which only solved 49.7% of the graphs. Following the bar chart in fig. 5.1a, MMAS achieved comparable results to the proposed ant algorithms in terms of being able to solve the problem instances. The overall performance of MMAS was however inferior in terms of solutions that had to be generated before finding the global optimum. In other words, MMAS required more time to solve the same flow problems. This is apparent from table 5.1. Nevertheless, MMAS was the best performing algorithm in one instance.

MMAS yielded slow convergence on the small-world graph set. Moreover, the slow convergence prevented MMAS from discovering the global optima within the iteration limit. The convergence of MMAS is rooted in the fine tuned balance between exploration and exploitation. This balance in governed by the definition of the ratio $\frac{\tau_{min}}{\tau_{max}}$ between the maximum and minimum pheromone level boundary. The decrease in performance as the graph size increased is striking when comparing the two bar charts in fig. 5.1. The small-world graph set contains larger graphs than the graph set created by Beasley [4], and it is apparent from the results listed in appendix A that MMAS struggle with the large graphs. The slow convergence may also be caused by the low pheromone evaporation rate. However, increasing the pheromone evaporation rate would in turn prevent the algorithm from finding the optimal solution in other graphs due to shifting of focus toward exploitation. Keep in mind that the result was generated by the best possible parameter configuration discovered through extensive analysis.

FT-ACO achieved good results in terms of finding the global optima in both graph sets. The combination of a fixed threshold labor division model and ant algorithm was able to solve 93% of the problem instances. This is a 16% increase over the benchmarking algorithm, MMAS. The results are even more interesting when studying the two graph sets separately. The two algorithms perform almost equally well on the graph set from Beasley [4], as can be deduced from the bar chart in fig. 5.1a. Nevertheless, FT-ACO yielded much better result on the small-world graph set than MMAS. The difference is obvious when comparing the bar charts in fig. 5.1. This is encouraging results, considering that many real world networks can be characterized as small-world [30, 41, 52].

Studying the table of results listed in appendix B, it is appreciable that in problem instances where both MMAS and FT-ACO converged to a suboptimal solution, FT-ACO had on average converged to a better solution than MMAS. Even though neither algorithm succeeded in persistently discovering the global optimum, MMAS appeared to be surpassed by FT-ACO.

FT-ACO is also converging faster than MMAS. The fast convergence in FT-ACO is a positive outcome of the labor division scheme. As described in section 4.3, FT-ACO will initially converge quickly and then initiate a exploration phase upon recognizing that is nearing stagnation. The outcome of the scheme is a search algorithm that quickly converges to a good solution, and in some instances, also the optimal solution. The experiment showed that FT-ACO on average converges faster than MMAS as exemplified by figs. 5.2 and 5.3. This is verifiable in the

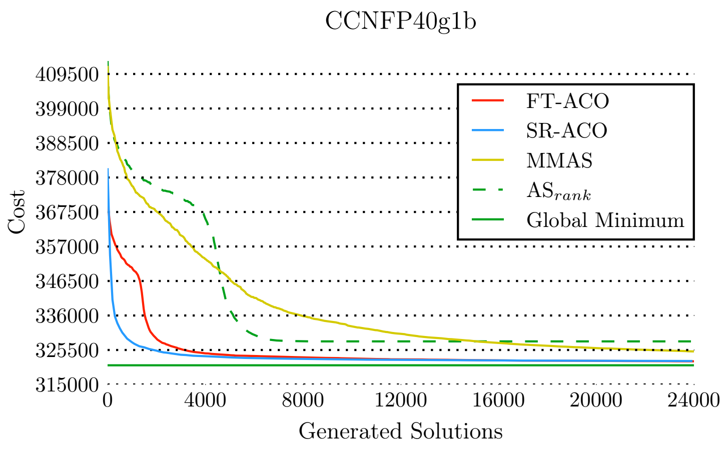


FIGURE 5.2: This graph exemplifies the situation where the search algorithms does not converge. SR-ACO trends toward having found the best solution of the competitors, while MMAS is bent towards producing the poorest solutions.

table of results listed in appendix A.

Nevertheless, FT-ACO was outperformed by Self-Reinforcement ACO. SR-ACO solved 98% of the test graphs. Even though the two labor division ant algorithms solved close to equally many instances, it is evident that SR-ACO required less solutions to be generated before discovering the global optimum when cross-referencing the results in appendix A. The superiority of SR-ACO is striking in the bar chart summarizing the small-world results, in fig. 5.1b.

SR-ACO performed very well and consistently found the optimal solution of all but 10 flow networks. It generally yielded fast convergence due to its exploitative nature. In addition to the fast convergence, SR-ACO appears to avoid stagnation very well. The search only entered stagnation in two percent of the test cases while the second best algorithm, FT-ACO, stagnated in four times as many problem instances.

SR-ACO appears to perform equally well on the graphs created by Beasley [4] and the small world graphs, which is apparent by comparing the bar charts in

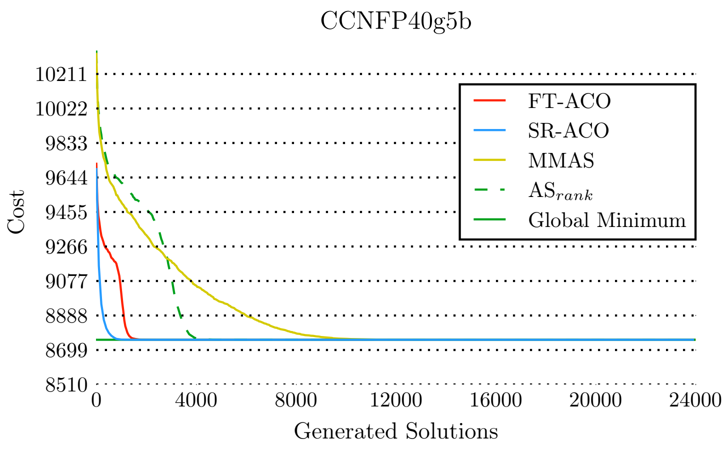


FIGURE 5.3: This graph exemplify the convergence rate of the ant algorithms in an easy problem instance. SR-ACO and FT-ACO quickly converge to the global optimum within a few iterations. Next follows the AS_{rank} algorithm, while the convergence rate of MMAS is slower due to the pheromone limiting scheme.

fig. 5.1a and fig. 5.1b. Like the other algorithms, group one and six in the graph set by Beasley [4] and group one in the small-world graph set appear to be the hardest to solve. All of these groups have high start-up costs relative to the unit costs in the flow network.

In general, it appears to be safe to say that SR-ACO outperforms both FT-ACO and MMAS. It is able to find the optimal solution for more graphs than MMAS and usually finds the optimum faster as well. SR-ACO appears to be able to avoid stagnation, without paying the price of slower convergence.

5.3.2 Search Dynamics

Max-Min Ant System

Two phases can be distinguished in the search process of MMAS. During the first phase MMAS generates solutions completely at random. During this phase, MMAS attempts to sample the entire solution space. Usually this phase is very short, only lasting a few iterations and can barely be perceived in fig. 5.4. However, the phase can be prolonged by adjusting the hand-tuned parameter Q, which affects the quantity of deposited pheromone $\frac{Q}{C_{\text{best}}}$ in MMAS. Remember that C_{best} is the cost of the best solution discovered.

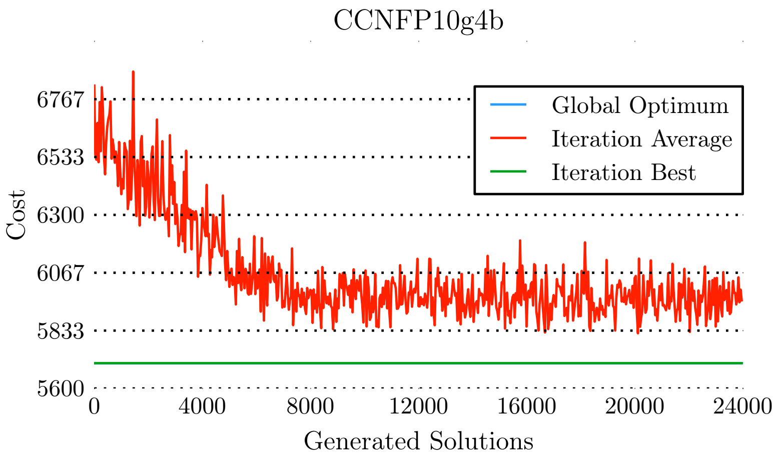


FIGURE 5.4: A single run of MMAS on *CCNFP10g4b*, which is considered an easy graph. The average value varies wildly as a result of the bounds on the pheromone levels. This jittering is intended to avoid stagnation.

The second phase of MMAS is also the final discernible phase. During this stage the algorithm continually optimize the global solution. The maximum and minimum bounds on the pheromone levels are in place to enforce the possibility to select an edge that hasn't yet been traveled. This phase is better illustrated in fig. 5.5.

As the cost of the best solution decreases, the minimum bound on the pheromone level also decreases. The decrease in the minimum bound, causes the average solution cost in the colony to increase after discovering new solutions with lower costs. This is observable in fig. 5.5. The increase in average solution cost occurs since the ants have become more likely to choose edges with lower pheromone strength. The average cost escalates until the system stabilizes with the new, lowered minimum bound.

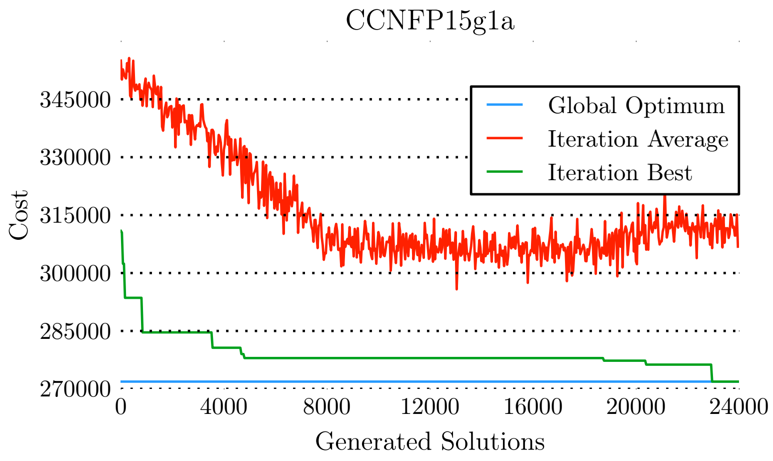


FIGURE 5.5: Illustrating the progression of a single run with MMAS on a difficult graph instance. The average value are varying wildly as a result of the bounds placed on the pheromone levels. The jittering is intended to avoid stagnating the search. An interesting observation in this graph is that the average solution quality of the population increase as the best solution S_{best} decrease in cost.

MMAS is metaphorically dependent on finding the correct hill to climb during the initial phase, when it still has the ability to jump around randomly. After the first phase has concluded, MMAS is increasingly more restricted to only look for solutions similar to the best solution it has discovered hitherto. The importance of finding the correct hill to climb is evident in fig. 5.6. The difference between the best and the worst run on this graph instance is discouraging. The worst run required over 14000 additional solutions to be generated before discovering the global optimum. This imposes the notion that MMAS will not perform well on graphs where the optimal solutions are situated in very peaky search landscapes.

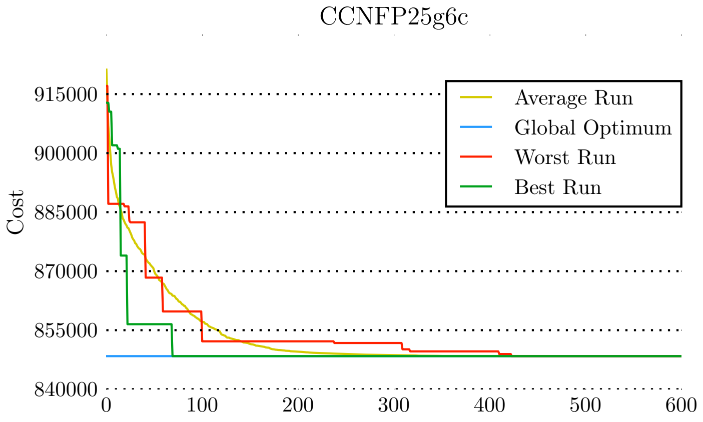


FIGURE 5.6: A graph showing the best and worst run of MMAS on CCNFP25g06c from Beasley [4]. Notice the difference between the two runs. The worst run required over 14000 additional solutions to be generated before discovering the global optimum.

In addition, this may also be part of the reason for why MMAS does not scale well to the large graphs in the small-world test set.

Fixed-Threshold ACO

The search procedure in FT-ACO consists of four recognizable phases as depicted in fig. 5.7. The first phase spans the initial iterations of the search procedure. During this phase, the pheromone levels on the edges are almost equal due to the initial pheromone quantity dispersed on the edges of the graph. Since the pheromone levels are uniform, the random generation of solutions will construct solutions from the entire search space. This phase is recognizable as a level population average in the discovered solution costs and observable in fig. 5.8.

The first phase concludes when the ratio in the pheromone levels between the edges that have received pheromones, and the ones that have not, reaches some pivot point. This pivot point symbolizes a state where the probability of constructing a solution with untraveled edges is very low. This second phase

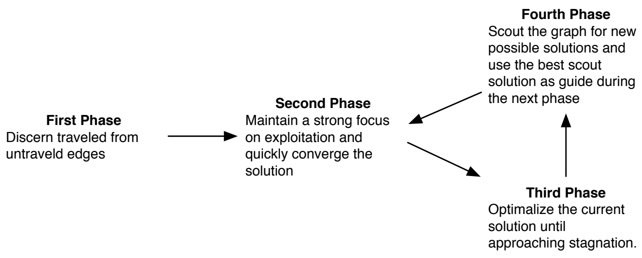

First Phase
Discern traveled from untraveld edges
Second Phase
Maintain a strong focus on exploitation and quickly converge the solution
Third Phase
Optimalize the current solution until approaching stagnation.
Fourth Phase
Scout the graph for new possible solutions and use the best scout solution as guide during the next phase

FIGURE 5.7: Illustrating the phases of FT-ACO. The final three phases recur until the search completed the given number of iterations K.

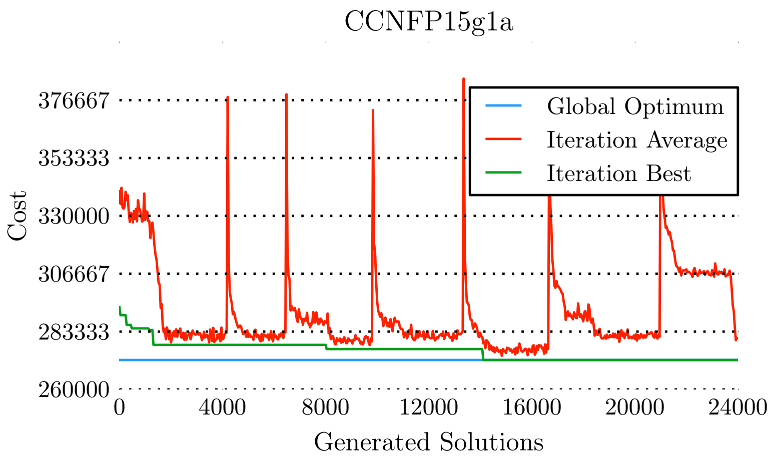


FIGURE 5.8: Illustrating the progression of a single run with FT-ACO on a difficult graph instance. The sudden increase in the population average indicates that the scouting stimulus reached its threshold and induced the ants to scout. The green curve indicate that the ants discovered better solutions through scouting after about 8000 and 14000 generated solutions.

is identifiable as a sudden drop in the average solution cost in the colony. The outcome of the phase is generally a good solution. If the problem instance is easy to solve, FT-ACO will often have produced the optimal solution upon terminating the second phase. An example of a easy graph problem is exemplified in fig. 5.9.

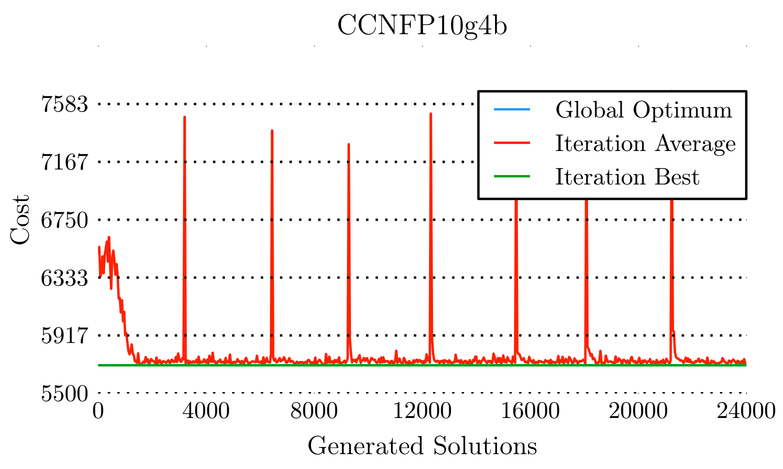


Figure 5.9: Illustrating the progression of a single run with FT-ACO on an easy graph instance. The sudden increase in the population average indicates that the scouting stimulus reached its threshold and in turn induced the ants to scout. In any case, FT-ACO discovered the global optimum without using the scouts.

The third phase is the optimization phase. This phase is characterized as when the colony no longer quickly finds better solutions. This phase has the same objective as the second phase, to exploitatively look for better solutions. The ant algorithm will approach stagnation during the third phase. To counteract the stagnation, FT-ACO employs the Fixed Threshold labor division model. The stimulus signal builds in strength when the ants no longer discover new solutions. This can be identified as a level plot line in fig. 5.8. When the stimulus has increased to a certain level the threshold function activates and FT-ACO enters the fourth phase.

The fourth phase is the scouting phase. This phase occur when the stagnation stimulus have reached some threshold value θ_{thresh}. During the course of the scouting period, the ants try to discover alternative solutions close to the best found solution S_{best} in the search landscape. This is achieved through a modulation of the edge selection mechanism as described in section 4.3.3. Subsequent to scouting in the fourth phase, the algorithm returns to the second phase.

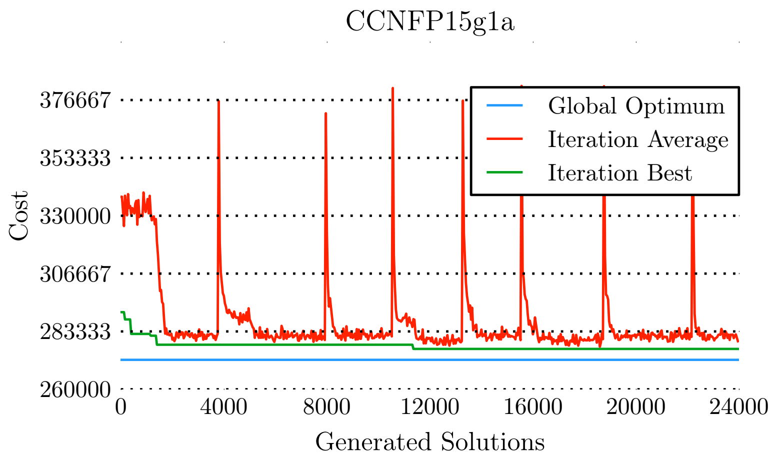


FIGURE 5.10: Illustrating the progression of a single run with FT-ACO on a difficult graph instance where the scouts did not provide the necessary guidance to find the global optimum. The sudden increase in the population average indicates that the scouting stimulus reached its threshold and in turn induced the ants to scout. Even though the scouts were issued seven times, the search did not find the optimum in this run.

FT-ACO relies upon the assumption that the scouts will eventually provide the necessary guidance such that the recurring second phase will lead to the global optimum. Unfortunately, FT-ACO did not always behave as intended. Intermittently the scouts would not provide the necessary guidance to find the global optimum despite trying multiple times as illustrated in fig. 5.10. Nevertheless, the inclination to quickly converge the search in order to issue a new scout epoch, both benefit the algorithm with fast convergence and balanced the exploration and exploitation.

Self-Reinforcement ACO

Two recurring phases can be identified during a search by SR-ACO: an exploitation phase and an exploration phase. Depending on the complexity of the flow network and length of the search, SR-ACO may switch between the two phases multiple times during a search.

The first phase is the exploitation phase, in which SR-ACO is able to find improved solutions repeatedly in subsequent iterations and several new solutions are added to the priority queue of unexplored solutions. As better solutions are found during the search, the length and frequency of the exploitation phase decreases. Because of this the exploitation phase is most apparent in the beginning of the search and can be observed in both figs. 5.12 and 5.13. An example of the exploitation phase reoccurring later can be seen near the termination of the search in fig. 5.13. The exploitation phase end and the exploration phase begin whenever the search reaches a local optimum.

At the termination of the exploitation phase, SR-ACO will have reached a local optimum. After this pivotal point, the search only discover a few solutions of the same quality as the best solution discovered yet S_{best}. The exploration phase can be observed as a plateau in the minimum cost, while the population average increase since the best solutions are being successively removed from the priority queue. As long as the search does not discover new solutions of the same quality as S_{best}, the colony's average cost will increase. This can clearly be seen in figs. 5.11 and 5.12. During the exploration phase, the search diverges further and further from the local optimum until hopefully a better solution is found. When an improved solution is discovered the exploration phase end and SR-ACO return to the exploitation phase. However, because some flow networks contain many solutions with similar costs, an increase in the colony average is not always as apparent. This is exemplified by fig. 5.13.

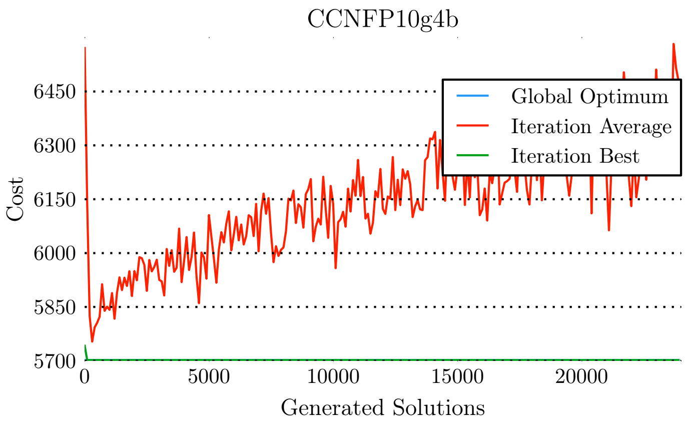


Figure 5.11: A single run by SR-ACO on an easy problem instance. The optimal solution is found almost instantly and one can observe how the population average increases as the search diverges from the optimum.

Although, SR-ACO initially has a uniform pheromone distribution similar to FT-ACO and MMAS, there is no discernible opening phase with an unchanging population average due to the high evaporation rate used by SR-ACO. The high evaporation rate immediately reduces the initial pheromone levels, so that the first reinforced solution is highly exploited. The lack of this phase can clearly be observed by comparing figs. 5.10 and 5.12, which respectively show a run by SR-ACO and FT-ACO on *CCNFP15g1a*.

The convergence curves in the previously listed figures clearly indicate that some of the flow problem instances were more difficult than others. Most of the figures referenced the same two graph from the problem set: CCNFP10g6a and CCNFP10g4a. A review of these two graphs, in addition to an overview of the problem sets, is presented the following section.

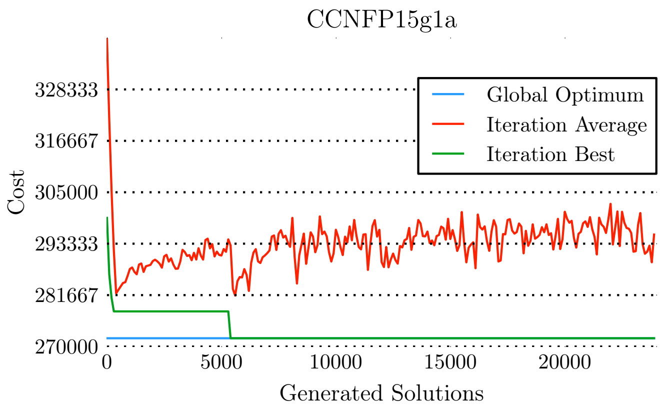


FIGURE 5.12: A single run by SR-ACO on a difficult problem instance. Notice that the population average initially decrease as the search discover solution improvements during the first few iterations. Following this period, the search is stuck for almost 5000 solution generations. During this phase SR-ACO examine successively poorer solutions in order to escape the local optimum. This is recognizable as an increase in the population average. Then, after 5000 generated solutions, SR-ACO escape the local optimum and found further solutions improvements. Once reaching the next optimum, after about 5500 generated solutions, the population average once again increase in order to escape the optimum. However, this was the global optimum and the population average can therefore be observed to trend upwards.

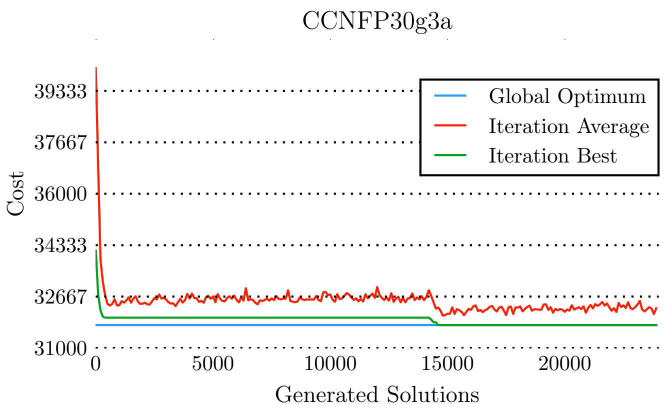


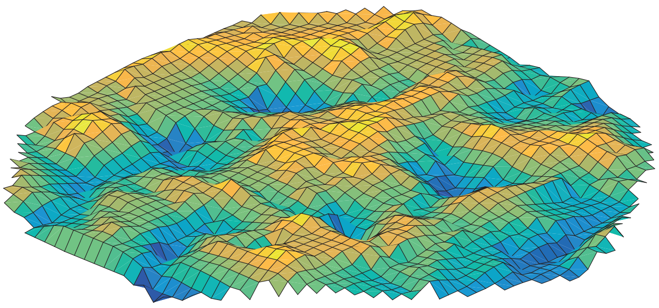
Figure 5.13: The figure illustrates a single run where SR-ACO quickly converge toward the best region of the search space. Nevertheless, the search spends a lot of time before discovering the global optimum.

5.3.3 Comparison of Flow Networks

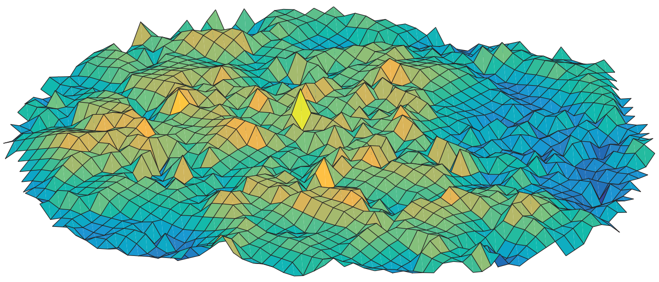
If one looks at the result tables in appendices A and B it is apparent that the most difficult flow networks have high start-up costs compared to unit-costs. This is consistent with the results of Monteiro et al. [33], who also found these groups to be the most difficult. A possible reason why these flow networks are more difficult, may be that the high start-up costs lead to similar solutions, in terms of common edges, having less similar costs.

Figure 5.14 shows 3D visualizations of the search landscapes for two flow networks; CCNFP15g1a that has very high start-up costs compared to the unit costs and CCNFP10g4 that has slightly higher start-up costs than unit costs. The search landscapes were created with t-SNE, a visualization method for high-dimensional data formulated by Maaten and Hinton [31]. From the figure it is apparent that CCNFP15g1a has a more rugged landscape than CCNFP10g4b. This may be an indication that similar solutions have less similar costs in CNNFP15g1a than in

CCNFP10g4b.



(A) A 3D visualization of the search landscape of CCNFP10g4b, which is an easy problem with slightly higher start-up costs than unit costs.

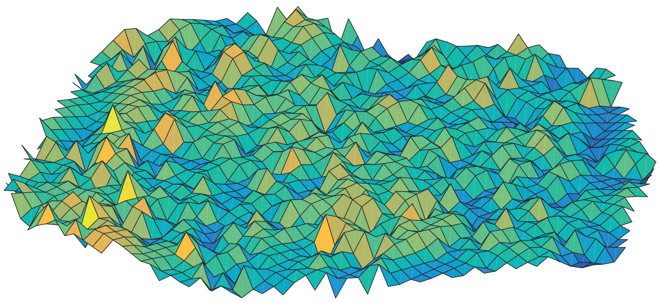


(B) A 3D visualization of the search landscape of CCNFP15g1a, which is a difficult problem with much higher start-up costs than unit costs.

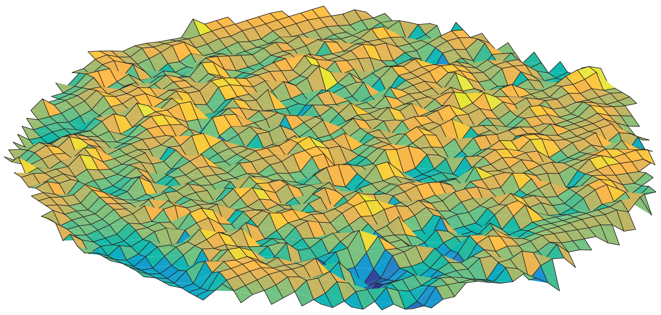
Figure 5.14: A comparison of the search landscapes of an easy and a difficult problem. Good areas are colored yellow, while bad areas are colored dark blue.

Another interesting observation that can be made is that the size of the solution space appears to have little effect on the difficulty of a flow problem. This is among others, the case with CCNFP10g6a and CCNFP10g4a. They are both tiny graphs with only ten nodes, but respectively have 592 and 124421 feasible solutions. Although all the algorithms always found the optimal solution for both problems, on average the algorithms required to generate more solutions to solve the smaller CCNFP106a than CCNFP10g4a. Furthermore, all of the algorithms required on average 2300 solutions or more to discover the optimum for

CCNFP10g6a, meaning that all the feasible solutions could have been generated almost four times if the algorithms instead enumerated all possible solutions. This implies that the algorithms generate some solutions several times over during a search.



(A) This is a 3D representation of the search landscape for the graph labeled CCNFP50g1a. In this representation, the goal is to hill climb to the brightest hill top. The landscape appears to be randomly scattered with local optima.



(B) This is a 3D representation of the search landscape for the graph labeled PowerClusterGraphN50g1c. In this representation, the goal is to hill climb to the brightest hill top. The landscape consists of larger ridge components with local optima.

FIGURE 5.15: This figure showcase projections of the search landscapes of CCNFP50g1a and PowerClusterGraphN50g1c. The two graphs are comparable in size, but are generated as a random and Small World graph respectively. It is apparent that the landscape in the random graph is more uniformly scattered with local optima.

In fig. 5.15 one can see 3D visualizations of the solution spaces for CCNFP50g1a and SmallWorldN50g1c, two graphs of equal size. The most apparent difference between the search landscapes is that SmallWorldN50g1c is more yellow, while CC-NFP50g1a is greener. This difference in color is an indication that CCNFP50g1a consist of more low-cost solutions than SmallWorldN50g1c. Except that the small-world network appear slightly smoother than the random network, the solutions spaces appear to be very similar. The similarity may be an indicator that it is the size differences and not the small-world property that primarily makes the small-world networks more difficult to solve.

When comparing the random flow networks by Beasley [4] to the small-world networks, one can see that MMAS and AS$_{rank}$ perform worse on the small-world networks than on the random networks, while SR-ACO and FT-ACO perform almost the same on the two graph sets. The difference in performance on the two graph sets may not necessarily be because of the small-world property but rather due to the size of the networks. The small-world set contains larger networks than the set by Beasley [4] and looking at the results in appendix A one can see that MMAS primarily struggles with the larger graphs. However, since the small-world effect only emerge in sizable graphs it was impossible to test MMAS on small-world networks of a size it handled well. The convergence of SR-ACO, FT-ACO and MMAS will be briefly discussed in the following section.

5.3.4 Convergence Rates

This section will discuss why the ant algorithms exhibit pronounced differences in their convergence rates. This is an interesting topic when establishing the best performing algorithm. A fast convergence rate in this setting implies that the algorithm requires few solutions to be generated before discovering the optimal solution. First, the convergence of Self-Reinforcement ACO (SR-ACO) algorithm will be discussed. Second, the focus is shifted to FT-ACO. Finally, the convergence of MMAS will be discussed. The observed convergence rates are typified by the graph in fig. 5.16.

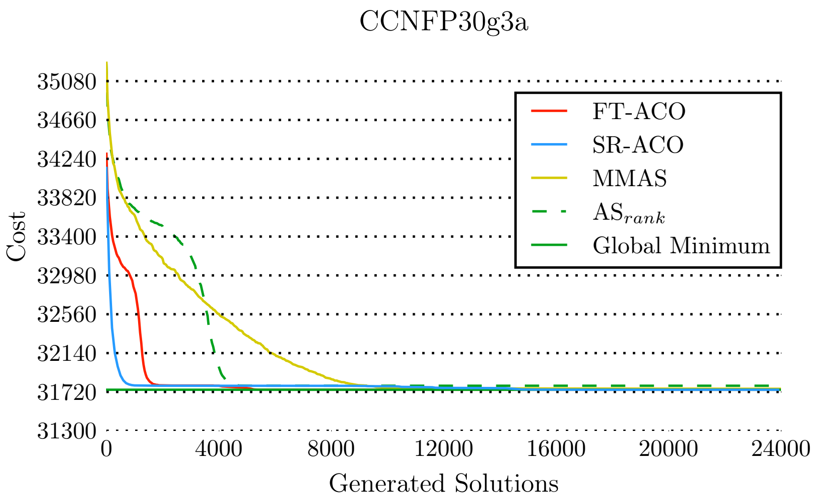


Figure 5.16: Illustrating the convergence of the algorithms on graph CCNFP30g3a. The curves represent the average cost of solutions and are produced from 100 runs with optimal parameter configuration.

SR-ACO was expected to exert a faster convergence than the competing algorithms due to the highly exploitative nature of the search procedure. SR-ACO is build upon the assumption that solutions of higher quality will be found in the vicinity of the currently best found solution. This intuition was also a perfect fit for the self-reinforcement model for labor division as described by Beshers and Fewell [5]. The focus on looking for solution improvements close to the optimal path is similar to how local search algorithms progress. Local search algorithms have been applied to MCFP before [54], but they are inferior in the problem domain. A local search algorithm constructs each solution in an isolated setting unlike SR-ACO, which also takes the previous states of the search into account. The success of SR-ACO, contrary to standard local search, may be related to the stigmergic information conveyed by the ants across the iterations of the algorithm. The importance of using the history of search states in ACO was pointed out by Stützle and Hoos [42]. The history of states is represented by the pheromones dispersed on the graph. As illustrated in fig. 5.16, SR-ACO converges very quickly toward the best regions of the solution space. However, the hunt for the exact global optimum resulted intermittently a prolonged search. This prolongation probably occurred if the search initially sloped down the wrong part of the search landscape as seen in fig. 5.13.

Unlike SR-ACO, FT-ACO share more similarities with traditional ant colony algorithms. FT-ACO relied on the stigmergic effect and that the pheromones deposited at iteration k would aid the ants in discovering a better solution in iteration $k + 1$. The fixed-threshold was only intended as a means to counter stagnation, contrary to additionally speeding up the search. The combination of a fixed threshold labor division model and ACO resulted in an algorithm that could converge quickly toward good regions of the solution space, but not necessarily the global optimum. The algorithm depended on the scouts to guide the search toward a region of the search space where the foragers could converge on the optimum solution. This seemed like an optimistic expectation. However, following the results listed in appendix A the scouts regularly provided the necessary guidance to promote the fast convergence. The expectation was facilitated by the dynamics of the threshold triggered scout phase. Every time the search approached a local optima, the colony would explore the landscape rather than getting stuck. Nevertheless, FT-ACO remained an algorithm that was very dependent on a successful scouting phase when solving hard problem.

Arguably, it is an advantage to have an algorithm that quickly converges toward good solutions. Fast convergence is preferable if the search may be terminated prematurely. Both FT-ACO and SR-ACO exhibited this dominance over MMAS.

MMAS clearly sported the slowest convergence rate, and consequently only outperformed the other algorithms in a single flow problem. MMAS tend to steadily discover solution improvements when searching, rather that having an exponentially decaying convergence as both FT-ACO and SR-ACO. However, following the definition of the pheromone limiting system, MMAS will only converge to local optima if the ratio $\frac{\tau_{min}}{\tau_{max}}$ was configured to a too small ratio.

Otherwise, the algorithm will always be able to generate unseen solutions, however seldom, since it is not improbable to choose an unseen edge. The result of this guarantee is that MMAS is better suited to optimize the last few percentages of a solution than FT-ACO, since FT-ACO would rather initiate a scout phase than being stuck trying to continually optimize.

As a final remark, it is important to note that the algorithms were optimized to find the best possible solutions by the means of generating 24000 solutions. In other words, none of the algorithms was configured to produce a steep slope in the convergence curve. Therefore, it is not correct to state that MMAS is a less efficient algorithm than its competitors. Nevertheless, it is interesting to observe that the proposed incorporation of labor division allow the ant algorithms to exercise faster convergence. This is of course advantageous if the algorithms were interrupted prematurely. Additionally, if they were applied in an industrial setting the search could swiftly provide the user with a decent solution while still searching for better candidates.

In order for FT-ACO to converge to a good solution in difficult search landscapes, it had to be able to spot the difference between only finding solutions of the same quality as S_{best} and stagnation. The following section presents a debate on how this differentiation could be accomplished.

5.3.5 Detecting Stagnation

Recognizing stagnation correctly was essential for FT-ACO. If the colony decided it was suffering from stagnation before it actually occurred, it would lack exploitation and be prevented from finding the optimal solution. Contrarily, if the colony spent time in a stagnated state this amounted to a waste of computational power. The following discussion is typified by the illustration in fig. 5.17.

Stagnation was defined in section 3.3 as the situation where all ants travel the same set of edges. Recognizing stagnation can therefore be achieved through comparing all constructed flow routes. The definition suggest that stagnation should be detected by calculating the difference between the solutions generated at iteration k and all previously discovered solutions $S_k^* \quad \forall k \in [1, K]$. Unfortunately, this would be computationally costly to perform in large problem instances.

Let M be the number of ants, K be the number of iterations and Z^0 be 0. The total number of comparisons Z calculated after K iterations is described by eq. (5.1).

$$Z^K = \sum_{k=1}^{K} M(Z^{k-1} + M - 1) \tag{5.1}$$

Measuring stagnation by comparing the solutions would require an exponential number of comparisons. Equation (5.1) tells us that after merely ten iterations with five ants, the check would have required more than 60 million comparisons. In addition, the complexity of calculating the comparison between two solutions scale linearly to the problem size. The stagnation recognition therefore calls for a heuristic approach. The rest of this section discusses three different approaches to heuristically detect stagnation.

The first technique was to focus on the solution cost and observe the decrease in the cost. This is a common approach in search, such as in genetic algorithms. The idea is that the search has converged when it no longer produce solutions with lower cost. This technique also appeared to be the most popular approach in ant colony optimization research. However, this did not perform well in the test since different flow paths in the graph could result in the same cost. These intermediate flow routing states of equal cost appeared to be essential when the colony required shifting their focus in the search landscape. As specified in the definition of stagnation, stagnation relates to the exploration of new solutions rather than the discovery of lower cost solutions. Arguably, the change in cost value is therefore a poor heuristic for stagnation.

The second approach to stagnation detection resembled the procedure discussed in the first paragraph. Rather than comparing the newly generated against all previous solutions, only a subset of the previously best discovered solutions are

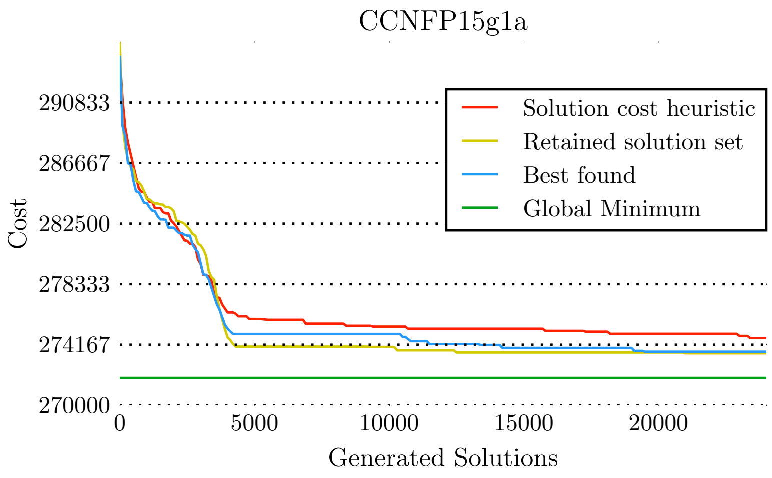


FIGURE 5.17: The curves exemplify the performance of the stagnation detection strategies discussed in section 5.3.5. The traditional solution cost heuristic (red) appear to be less suitable than the stagnation evaluation approaches (yellow and blue). The yellow line represents a search where the entire history of constructed solutions is retained. The blue line denotes a search where the stagnation level is calculated only measuring the difference between the iteration solutions and the best solution S_{best}. This problem instance was one on the most difficult in the two test sets.

retained. The required set size depends on the problem difficulty. More challenging graphs require additional solutions to be retained. Additionally, the larger the set, the more accurate would the stagnation measurement become.

The third technique is an extension of the second approach where the set of retained solutions has size one. In other words, only the best observed solution is retained. The stagnation recognition procedure would thereby compare all newly generated solutions to the best found solution. This appeared to yield a sufficient between the computational requirement of calculating the comparison, and the accuracy of the stagnation measure.

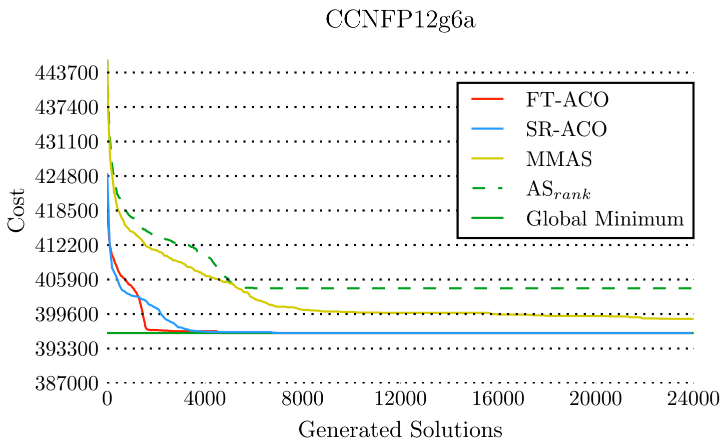
The essence of these observations was that only observing the fitness value when determining stagnation, proved to be inadequate. The best results were, as expected, obtained when using the comparison approach and retaining all of the previous solutions. However, satisfactory results were achieved when using only the globally best found solution as is typified in fig. 5.17.

The following section outlines the stagnation avoidance efficiency of MMAS, SR-ACO and FT-ACO. The results it presents provide some interesting insights on the difference between the seemingly most popular stagnation avoidance method in literature, MMAS, and the proposed algorithms.

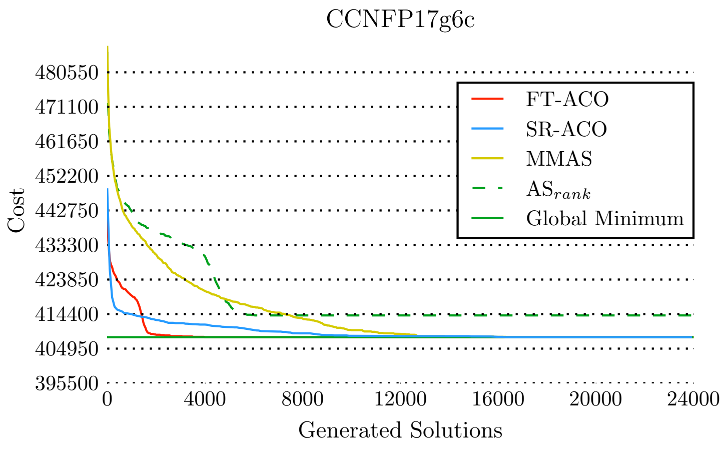
5.3.6 Stagnation Avoidance Capabilities

This section will discuss the stagnation avoiding capabilities of Max-Min Ant System (MMAS), Fixed-Threshold ACO and Self-Reinforcement ACO, respectively. The aim of this section is to shed light on which problem configurations the algorithms was more or less likely to encounter stagnation. Finally, the two proposed labor division approaches will be compared to MMAS.

All the stagnation-avoiding ant algorithms clearly outperformed AS_{rank} on the test sets. AS_{rank} failed to solve 151 MCFPs, over half of the problem instances. The algorithm was included to demonstrate whether early convergence was an issue in this problem domain, and the experiment results of AS_{rank} proved so.



(A) Graph CCNFP12g6a: The curves show that FT-ACO and SR-ACO quickly converge to the global optimum, while MMAS is starting to show indication of stagnation. However, all of the stagnation avoiding algorithms perform considerably better than AS_{rank} which does not incorporate stagnation avoidance.



(B) Graph CCNFP17g6c: The stagnation avoidance schemes enable the search to reach the global optimum. Remember that AS_{rank} does not incorporate stagnation avoidance.

FIGURE 5.18: Comparison of how the different stagnation avoidance schemes perform when AS_{rank} stagnate.

Comparably, MMAS performed well and stagnated in only 42 cases. Moreover, the stagnation tally of MMAS may be pessimistic since MMAS will always find the optimal solution given infinite iterations. This guarantee originates from the observation that no edges will ever be impossible to select in MMAS. However, for the benefit of the experiment it was necessary to set a limit on the number of iterations that could be carried out.

The progression of MMAS on problems where it either stagnated or did not converge typically had the outline as in fig. 5.18a. The curve representing MMAS show that the search leads of on a steady improvement for each iteration until some point where it flattens out. At this stage, it has become unlikely for the ants to traverse edges that do not belong the set of edges in the optimal solution due to the differences in pheromone levels. From this point onward, the search appears only to discover new solutions through strokes of luck.

The result from the stagnation experiment listed in appendix B also indicated that MMAS performed progressively worse as the graphs grew in size. MMAS appeared to be strained on the large small-world instances, solving less than 50% of the MCFPs in this test set.

FT-ACO stagnated in 21 MCFP instances, halving the number of stagnated flow problems compared to MMAS. However, the occurrences of stagnation in FT-ACO and MMAS only overlapped in eleven instances. In the remaining ten cases where only FT-ACO stagnated, MMAS had not yet converged to a solution. This resulted in a situation where FT-ACO had stagnated on suboptimal solutions that were in turn better than the unconverged result from MMAS.

In four cases where FT-ACO stagnated, both MMAS and SR-ACO was able to solve the problem. This confirms that FT-ACO did not dominate the results from MMAS. More precisely, FT-ACO rather proved to be a more efficient solver in general than MMAS in the setting of this experiment.

SR-ACO stagnated on a suboptimal solution in six test cases, 85% less than MMAS. This is a striking improvement in terms of ameliorating the stagnation problem in ant algorithms.

In the six the problem instances where SR-ACO failed to discover the optimal solution, all of the other algorithms failed to converge on the global optima as well. Thereby, making SR-ACO the experiment's most reliable stagnation avoidance scheme. In addition, SR-ACO proved to be a peerless algorithm in high start-up cost problems, as interpretable from the results in appendix A.

Stagnation generally occurred in the flow problems where the start-up cost was much larger than the unit cost. This may be crucial in industrial applications, where the start-up cost tend to be considerably larger than the unit cost. Creating a transportation office will always be costlier than sending a single letter.

The conditions of the experiment were set up to constitute a justifiable setting through optimizing each of the algorithm's parameters to the problem sets, as outlined in section 4.4.7. Under these conditions, SR-ACO clearly outperformed the rest and only stagnated on suboptimal solutions in ten of the test cases.

The subsequent sections analyze interesting aspects of applying Ant Colony Optimization to the MCFP domain. These sections are not directly related provide answers to the research questions, but constitute important intuitions for solving MCFP with ACO.

5.3.7 Similarities Between Good Solutions

The search of an ACO algorithm is guided by pheromones. When the pheromones evaporate and new ones are deposited on the edges in a solution, it makes the reinforced edges more likely to be selected during solution construction in the next iteration. The extent a solution is reinforced depends on the evaporation rate, the quantity of pheromones deposited and potential pheromone bounds. In order for an ACO algorithm to exploit the pheromones it is necessary for similar solutions, measured by the number of common edges, to also have similar cost. However, the search of an ACO algorithm is complex. The similarity between the reinforced solution and the newly generated solutions can either be high or low depending on the pheromones, and will vary throughout the search. This is a major difference between ACO and traditional local search, where the similarity between the current solution and its generated neighbors is often constant and high throughout the entire search.

If one looks at a set of the n best solutions for a graph, one can both see that many edges are shared among the solutions and some edges are not used by any of the best solutions. This is illustrated in fig. 5.19, where the frequency of edges among the 100 best solutions for *CCNFP10g01c* is shown. These solutions were obtained by enumerating the entire solutions space of this flow problem. From fig. 5.19 it is discernible that some edges are present in most of the solutions. These edges appear as dark lines in the figure. One of the edges even appears in all but one of the 100 solutions. This is an indicator that solutions with similar costs share edges, which is necessary for ACO to operate in the MCFP domain.

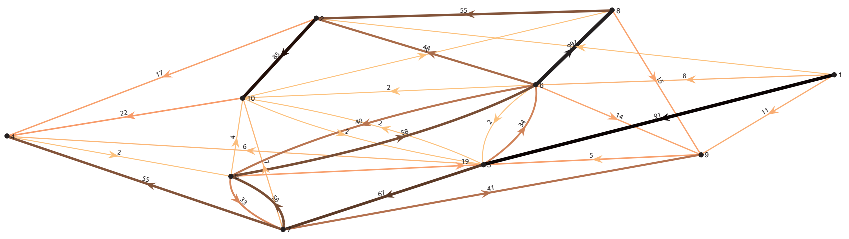


Figure 5.19: This plot show the number of times an edge was included in the routes of the 100 best solutions for *CCNFP10g01c*. Frequent edges are thicker and darker while less frequent edges are thinner and lighter. The observation that good solutions share edges is important in order for an ACO algorithm to operate efficiently in the MCFP domain.

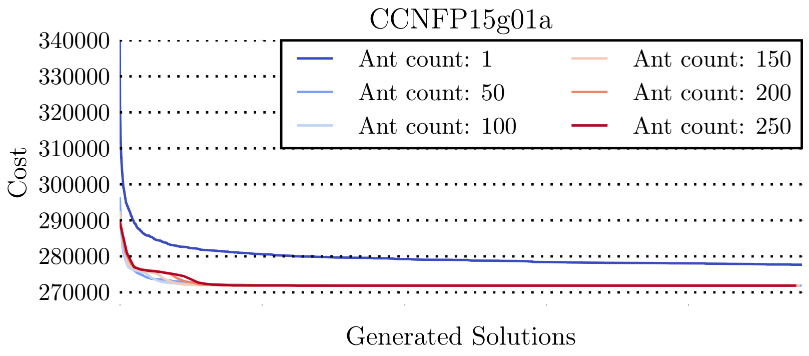
5.3.8 Balancing Ant Count and Iterations

Figure 5.20 show how different combinations of ants and iterations affected the convergence of SR-ACO and FT-ACO. The number of solutions generated, that is the product of ants and iterations, was kept constant for all of the combinations. Thus, the amount of work performed and time spent by all the combinations were the same. The effect of manipulating the number of ants is interesting to study, since the user of the algorithm must configure this parameter.

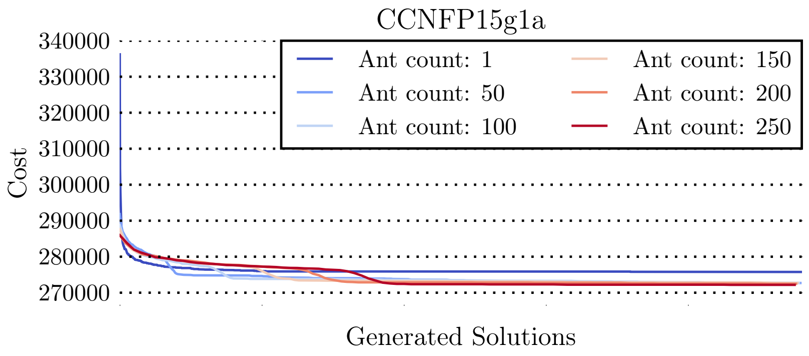
The degree of communication through stigmergy is however *not* equal among the different runs. Runs with fewer ants, are inherently subjected to a stronger stigmergy effect since they are allowed to carry out more iterations, depositing more pheromone on the edges.

From fig. 5.20a it is apparent that SR-ACO is not sensitive to this balance. The ant count in SR-ACO is directly related to the exploration focus. If the number of ants is increased, SR-ACO will focus more on exploration since more solutions will be generated in the neighborhood of the reinforced solution. From fig. 5.20a it is perceptible that using around 100 ants was optimal for this graph. While in general, SR-ACO performed best with 140 ants.

In contrast, FT-ACO was more sensitive to this parameter. The balance between the number of ants and the number of iterations executed has a complex effect on FT-ACO. Applying many ants to a problem entail more exploration, as in SR-ACO. In addition, it also implies that the search will approach stagnation faster since the same solutions will be generated more often when there are more ants. However, when the search approach stagnation the threshold function would trigger scouting and the colony perform better with more scouts. In general, the most suitable configuration for FT-ACO was 40 ants and 600 iterations. Arguably, it appears to be important to balance the number of ants and the number of iterations in order to balance exploration and exploitation. Too few ants can lead to excessive exploitation and convergence to local optima. On the other hand, too many ants can lead to excessive exploration and too slow convergence.



(A) The effect of different combinations of ants and iterations on the convergence of SR-ACO. The number of ants applied to the search in SR-ACO directly affect the degree of exploration performed in the proximity of each reinforced solution. As long as the ant count is "high enough", the algorithm will succeed.



(B) The effect of different combinations of ants and iterations on the convergence of FT-ACO. The more ants applied to a problem, the more efficient will the exploration phase become, since the colony would be able to generate a larger set of exploratory solutions.

Figure 5.20: The effect of different combinations of ants and iterations on the convergence of SR-ACO and FT-ACO.

5.3.9 Heuristic Value in the Deposited Pheromone Levels

As described in section 4.3.5, the quantity deposited in each iteration is equal to the ratio of improvement in the solution cost. Thus if the best solution S_{best} remains identical between two iterations, only one pheromone unit will be deposited. However, if the colony discovers a solution with half the cost in the next iteration, the edges in the best solution will be reinforced with two pheromone units. The quantity of pheromones deposited is therefore reflected in the relative improvement of the solution cost. A reduction in solution cost indicates that the ants are approaching an area of the search landscape where the optimal solution may reside.

The idea of proportionally rewarding ants who discover better good solutions follow the analogy in nature where some ant species, such as the *Iridomyrmex humilis*, show food preference [3].

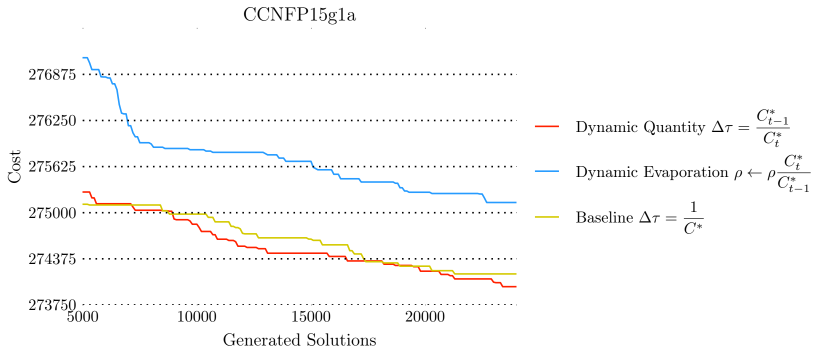


FIGURE 5.21: A graph exemplifying that depositing pheromone relatively to the solution improvement offer a slight improvement compared to the standard $\frac{1}{C^*}$ where C^* is the best found solution. The blue line depict how the dynamic evaporation scheme performed. Remember the ρ is the evaporation factor, and that $1-\rho$ pheromones are conserved between iterations. The curves represent the average cost of solutions and are produced from 100 runs with optimal parameter configuration.

We also tested the hypothesis that dynamically adjusting the deposited pheromone quantity might produce the same behavior as increasing the evaporation rate ρ for one iteration. This followed the intuition that evaporating the pheromones from the previous iteration makes the colony more likely to follow the route of the new solution. The experiment showed that evaporating the pheromones more when discovering cheaper solutions produced a much worse result than using the dynamical quantity strategy. This is illustrated in fig. 5.21. The decline in performance may arise from the side effects of evaporating large quantities of pheromones. By removing a lot of pheromones quickly, the edges with scant pheromone would be nearly impossible to travel and thus only a portion of the search landscape would be reachable.

Consequently, the idea to reward solution improvements can be considered a suitable method to guide the colony towards newly discovered regions in the search landscape.

5.4 Summary

In this chapter we have described the experiment results. First, a general overview of the results was presented. We saw that the performance of the well-established Max-Min Ant System dropped rapidly as the graph size increase. The result also indicated that the SR-ACO was the fastest solver in terms of solutions that had to be generated before the global optimum was discovered.

Finally, this chapter illustrated how the stagnation avoidance schemes in the different algorithms safeguarded the search from early convergence. We saw how the scouts in FT-ACO were issued by the Fixed-Threshold model, how MMAS used pheromone boundaries and how the priority queue in SR-ACO enabled the ant colony to explore different parts of the search land scape.

Additionally, the experiment showed that SR-ACO and FT-ACO clearly outperformed MMAS in the large small-world graphs. This is an important observation since most realistic network flow problems typically has the small-world property.

The following chapter will provide an comprehensive interpretation of the achieved results. Finally, the report wraps up with a brief mention of extensions that could build upon the foundations of this thesis.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The essential constituent of ant algorithms is pheromone. The pheromones allow the ants to communicate through stigmergy. In analogy to ants in nature, stigmergy is employed to establish ant trails. The upshot of this thesis is that using stagnation avoidance inspired by naturalistic ants forms a feasible strategy to avoid stagnation. The results from the experiments did however also indicate that augmenting ant colony optimization with labor division did not produce a flawless system.

Similar to the results of the Fixed-Threshold ACO algorithm, labor division in nature does not provide complete protection from stagnation. Naturalistic stigmergy is also fallible and trail following species are vulnerable to end up following their own tracks. This path movement is known as circular milling, and was first observed in 1896 by Jean-Henri Fabre in the caterpillar *Cnethocampa pityocampa* [40]. Milling has also been observed in some ant species, such as Army ants. These ants blindly follow the trail of pheromones laid out before them, and circular milling is capable of bringing entire colonies to an end. The milling occurs when the pheromone trail is sufficiently reinforced, so that no colony member will deviate from the circular movement. Consequently, the entire colony will continue to walk until dying of exhaustion.

Despite the weaknesses in stigmergic communication, insects have thrived on this planet for millions of years. Some of their success is attributable to social organization. Similar to the success of organization in nature, introducing a simplified labor division to artificial ant colony optimization improved the evolution of solutions.

Division of Labor This thesis aimed to study the two questions posed in the introductory chapter. First, we set out to examine how labor division could be combined with ant algorithms and whether this affected the search. The first question was:

1 *How can labor division be incorporated into ACO to counter stagnation and how does it affect the performance?*

This topic lead us to evaluate two different labor division models from the field of biological sciences. While the viability of using division of labor to avoid stagnation could be deduced from the experiment results, the performance aspect has to be juxtaposed to how it affected the balance between exploration and exploitation.

The two variations on how to incorporate labor division into ant colony optimization represent divergent ideas of modeling labor division. The first approach was to model the artificial colony as a system consisting of two distinct castes. This caste system was closely related to the division of labor observed in naturalistic ants [14, 48, 49]. This was achieved by establishing behavioral differences between foragers and scouts. These two castes were separated by weighting their local transition probabilities differently during trail construction. In terms of avoiding stagnation, this scheme resulted in a self-organized system that was more resilient compared to the popular ant algorithm MMAS.

The second approach to incorporate labor division took a more holistic approach. In this approach, the balancing of exploration and exploitation was not considered on the individual level. Instead, the colony as a whole would achieve this balance through selectively concentrating the search effort in regions of the landscape where good solutions were discovered. The self-reinforcing behavior was effectuated by only trying to generate low cost solutions. This behavior entailed that the Self-Reinforcement ACO search was very exploitative, yet it resulted in the most resilient ant algorithm in this study.

Balancing Exploration and Exploitation After we had established that incorporating the labor division models was a feasible approach to avoid stagnation, the focus shifted to analyze how they affect the search efficiency. The second research question was:

2 How are exploration and exploitation in ACO balanced by the Fixed-Threshold and Self-Reinforcement labor division models?

The stagnation problem in ant algorithms is another demonstration of the importance of balancing exploration and exploitation in an informed search. If the artificial ant colony stops exploring the environment before they have located the optimal solution, the search can be described as too exploitative. This situation arises when the pheromone levels on a set of suboptimal edges become so strong that the entire colony always chooses to traverse these edges. Apparently, the most popular technique to tackle this challenge is to limit the strength of all pheromone signals by introducing upper and lower bounds. However, since this tactic directly manipulates the pheromone trails, it also interferes with the effectiveness of stigmergy. The predominant disadvantage of utilizing bounds is the ensuing reduction in convergence rate. The slower the convergence rate, the more iterations are consumed before converging on the global optimum. Following this rationalization, it would be favorable to avoid interfering with the stigmergic process.

The Fixed-Threshold ACO algorithm did not impose the same strict boundaries on the pheromone level as MMAS. This led to a much faster convergence than what was observed in Max-Min Ant System. Rather than to guarantee exploration through hampering strong stigmergic signals, the FT-ACO search would instead issue explorers when the ants no longer discover new routes. The Fixed-Threshold model governed the balance between exploration and exploitation by observing the level of stagnation. When the need for exploration increased rapidly, the colony may decide to initiate an exploration phase. This strategy also implies that FT-ACO is very dependent on the scouts to guide the search toward regions of the search space where the foragers may discover the optimal solution. Nevertheless, the experiments indicated that the scouts succeeded in the majority of trials.

The labor division model utilized in Self-Reinforcement ACO was more intertwined. The Self-Reinforcement model itself is a conceptual hypothesis on how division of labor may occur in real life. The notion that an agent would keep performing a task if they perceived that the activity yielded a positive reward seems intuitively appealing. The idea was realized in the ant algorithm by looking at the colony as a whole. Rather than reinforcing an ant to move along the same path as the previous iteration, the colony was reinforced to forage for solutions that contained edges from the best discovered solutions. This behavior was implemented using a priority queue of solutions to reinforce. SR-ACO balanced exploration and exploitation through adjusting the size of the priority queue. The queue would always be updated with the most recent, best solutions. The smaller the queue, the more likely would it be to append recent solutions to the queue. Thus, forcing the search to progress toward different regions of the search space. Exploration was in other words a co-product of retaining a small priority queue.

The level of exploration in SR-ACO was also governed by the size of the colony. The number of ants in a colony was directly proportional to the number of routes generated before a solution was removed from the queue. In problem instances where exploration was important, SR-ACO favored a large colony rather than carrying out many iterations.

Algorithmic Performance The final topic emphasized in thesis was search performance. The goal was to identify if division of labor either augmented or impeded the dynamics of an ACO search. This subject matter was covered in the first research question, but could only be answered after having observed how the proposed algorithms balanced exploration and exploitation.

1 *How can labor division be incorporated into ACO to counter stagnation and how does it affect the performance?*

Incorporating division of labor yielded faster convergence than manipulating pheromone signals in most of the test. MMAS required typically more than twice the number of solutions to be generated before discovering the optimal solution. This indicates that pheromone limitation may not be the most efficient approach to prevent stagnation. Arguably, the results indicate that the performance of FT-ACO and SR-ACO surpassed MMAS.

However, both of the proposed algorithms introduced computational overhead. FT-ACO imposed the necessity to evaluate the level of stagnation during the progression of the search in order to abide by the Fixed-Threshold Model. If the complexity of measuring the stagnation level dominates the complexity of calculating a solution, the benefit of FT-ACO might be defeated. In similar fashion, the SR-ACO algorithm requires additional bookkeeping to maintain its priority queue. Consequently, the proposed algorithms are ill suited to outperform MMAS in problem domains where the complexity of constructing solutions are low. However, many industrial applications tend to operate on very large problem instances and this may favor SR-ACO and FT-ACO. The labor division augmented algorithms also outperformed MMAS on the test set with small world graphs. Arguably, this strengthens the impression that division of labor may be a more useful stagnation avoidance scheme in realistic applications.

In addition to the increased computational overhead in FT-ACO, the algorithm also required the user to hand-tune on more parameter than in MMAS. While MMAS require the user to specify the estimated likelihood of constructing the optimal solution p_{best}, FT-ACO required the user to specify both an exploration factor γ, that regulate how exploratory scouts will be, and the threshold value θ regulating when the stagnation stimulus should trigger exploration. The introduction of more parameters is unpopular among programmers. However, following extensions to the Threshold-Model, the θ parameter may be possible to learn through exposing the colony to problem instances over many iterations [9].

Synopsis The aim of this thesis was to investigate how division of labor could be integrated in artificial ant colony optimization. The two techniques in this thesis for incorporating social organization resulted in a different take on managing stagnation than the conventional pheromone limitation. The two models appeared to be more capable of solving large graphs than the benchmarking algorithm, MMAS. Nevertheless, the results were obtained from only a single problem domain. Therefore, it is not justifiable to conclude that MMAS was inferior. Most importantly, we have shown that division of labor is a viable countermeasure to stagnation in ant colony optimization.

6.2 Future Work

6.2.1 Problem Domain

While the FT-ACO and SR-ACO algorithms performed comparatively well in the minimum cost flow problem domain with concave cost function, they must be evaluated in further domains before we can justify that division of labor is more efficient than MMAS. We prioritized to assess the algorithms extensively in a single problem domain rather than superficially comparing them across a handful of optimization problems. As a result, we are able to conclude that MMAS is inferior in the MCFP domain. However, it still remains to investigate whether this conclusion can be generalized to other problem domains.

6.2.2 Combining SR-ACO and FT-ACO

The two labor division models studied in this thesis are not mutually exclusive. Instead, they may be combined to form an even more resilient system. SR-ACO featured the quickest convergence, but was also prone to plateauing in the most difficult problem instances. A Fixed-Threshold Model similar to the stagnation handling in FT-ACO could maybe detect these plateaus. Merging these two techniques could therefore result in an even better solver. If the scouts issued from FT-ACO could provide the foragers in SR-ACO with solution alternatives beside those in the priority queue, this would supply SR-ACO with an additional tool to avoid stagnation. However, a synthesis of these two algorithms would also incur more computational overhead. Nevertheless, a combination of the two algorithms may result in a strategy capable of harvesting the virtues of both algorithms.

Appendix A

Convergence Experiment Results

Table A.1: This table present the results of Experiment A: Convergence. The first column *Avg.Dev* list the deviation between the average solution and the global minimum. A value of 0.5 imply that the average discovered solution cost was 1.5 times the minimum cost. In other words, closer to 0 is better. The second column list the standard deviation in the cost obtained by the given algorithm. The third column list the number of iterations that was required before the algorithm discovered the global optimum. Every instance was solved 100 times with optimal parameter configuration. The algorithms were allowed to generate at most 24000 solutions. A '–' in the *Avg.Dev* column indicate that the algorithm always converged to the global optimum.

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP10g01a	–	1080	–	**900**	–	3080
CCNFP10g01b	–	1040	–	**400**	–	1960
CCNFP10g01c	–	**1320**	–	1600	–	4320
CCNFP10g02a	–	**160**	–	200	–	600
CCNFP10g02b	–	360	–	**300**	–	920
CCNFP10g02c	–	400	–	**300**	–	600
CCNFP10g03a	–	3720	–	**1700**	–	3000
CCNFP10g03b	–	880	–	**300**	–	1640
CCNFP10g03c	–	600	–	**300**	–	1480
CCNFP10g04a	–	960	–	**500**	–	2600
CCNFP10g04b	–	240	–	**200**	–	280
CCNFP10g04c	–	560	–	**300**	–	1880

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP10g05a	–	21080	–	**800**	–	4040
CCNFP10g05b	–	1040	–	**700**	–	3520
CCNFP10g05c	–	840	–	**300**	–	1480
CCNFP10g06a	–	12720	–	**2300**	–	3000
CCNFP10g06b	–	840	–	**800**	–	2200
CCNFP10g06c	–	1400	–	**500**	–	3920
CCNFP10g07a	–	1160	–	**400**	–	1640
CCNFP10g07b	–	**200**	–	300	–	760
CCNFP10g07c	–	**1040**	–	1700	–	4120
CCNFP10g08a	–	960	–	**400**	–	2720
CCNFP10g08b	–	600	–	**500**	–	1880
CCNFP10g08c	–	840	–	**300**	–	1960
CCNFP10g09a	–	**160**	–	200	–	400
CCNFP10g09b	–	1080	–	**400**	–	1440
CCNFP10g09c	–	**400**	–	700	–	1840
CCNFP10g10a	–	1120	–	**500**	–	1880
CCNFP10g10b	–	1040	–	**300**	–	1800
CCNFP10g10c	–	920	–	**300**	–	600
CCNFP12g01a	–	**680**	–	1600	–	3120
CCNFP12g01b	–	3200	–	**2700**	–	7240
CCNFP12g01c	1.49e-4	–	–	**18000**	1.48e-5	–
CCNFP12g02a	–	1160	–	**300**	–	2280
CCNFP12g02b	–	1200	–	**500**	–	4840
CCNFP12g02c	–	3880	–	**700**	–	5360
CCNFP12g03a	–	1160	–	**500**	–	3360
CCNFP12g03b	–	9640	–	**3800**	–	6960
CCNFP12g03c	–	**200**	–	300	–	360
CCNFP12g04a	–	800	–	**500**	–	1560
CCNFP12g04b	–	280	–	**200**	–	720
CCNFP12g04c	–	3520	–	**800**	–	4920
CCNFP12g05a	–	1080	–	**700**	–	3640
CCNFP12g05b	–	1000	–	**300**	–	1640
CCNFP12g05c	–	1040	–	**600**	–	4400
CCNFP12g06a	–	**4680**	–	7100	6.52e-3	–
CCNFP12g06b	–	20600	–	**8800**	–	14760
CCNFP12g06c	–	2120	–	**2000**	–	8000
CCNFP12g07a	–	600	–	**300**	–	1400
CCNFP12g07b	–	19240	–	**2800**	–	8360
CCNFP12g07c	–	1280	–	**800**	–	4720
CCNFP12g08a	–	7520	–	**1600**	–	5440
CCNFP12g08b	–	1160	–	**1100**	–	3680

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP12g08c	–	1080	–	**400**	–	1760
CCNFP12g09a	–	1400	–	**600**	–	1400
CCNFP12g09b	–	1320	–	**700**	–	4640
CCNFP12g09c	–	1520	–	**400**	–	1720
CCNFP12g10a	–	1680	–	**900**	–	2800
CCNFP12g10b	–	1360	–	**700**	–	3800
CCNFP12g10c	–	1080	–	**300**	–	1760
CCNFP15g01a	1.62e-3	–	–	**5500**	8.06e-3	–
CCNFP15g01b	–	1360	–	**400**	–	3240
CCNFP15g01c	1.24e-3	–	–	**6700**	–	21480
CCNFP15g02a	–	**1480**	–	7900	–	5000
CCNFP15g02b	–	1440	–	**700**	–	5080
CCNFP15g02c	–	1680	–	**1000**	–	7280
CCNFP15g03a	–	1200	–	**800**	–	4040
CCNFP15g03b	–	1360	–	**1000**	–	5560
CCNFP15g03c	–	1040	–	**300**	–	2120
CCNFP15g04a	–	1720	–	**1200**	–	5440
CCNFP15g04b	–	1480	–	**800**	–	3840
CCNFP15g04c	–	1080	–	**500**	–	2640
CCNFP15g05a	–	1520	–	**600**	–	5800
CCNFP15g05b	–	920	–	**300**	–	1960
CCNFP15g05c	–	1280	–	**900**	–	4800
CCNFP15g06a	–	**1440**	–	1900	–	5360
CCNFP15g06b	–	**5800**	–	6500	–	20240
CCNFP15g06c	–	10520	–	**3500**	–	13560
CCNFP15g07a	–	9080	–	**4300**	1.37e-3	–
CCNFP15g07b	–	1040	–	**300**	–	3320
CCNFP15g07c	–	1760	–	**700**	–	4640
CCNFP15g08a	–	1440	–	**600**	–	8680
CCNFP15g08b	–	1240	–	**600**	–	3760
CCNFP15g08c	–	1360	–	**600**	–	5160
CCNFP15g09a	–	2280	–	**1200**	–	5720
CCNFP15g09b	–	1560	–	**700**	–	4600
CCNFP15g09c	–	1600	–	**800**	–	5280
CCNFP15g10a	–	2040	–	**1600**	–	5000
CCNFP15g10b	–	1640	–	**700**	–	3800
CCNFP15g10c	–	1160	–	**500**	–	2200
CCNFP17g01a	1.05e-3	–	9.59e-5	–	–	**13920**
CCNFP17g01b	1.57e-6	–	–	**11800**	1.93e-4	–
CCNFP17g01c	–	1440	–	**1000**	–	4880
CCNFP17g02a	–	1240	–	**500**	–	2760

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP17g02b	–	1320	–	**700**	–	4920
CCNFP17g02c	–	1920	–	**1200**	–	9960
CCNFP17g03a	–	960	–	**400**	–	3520
CCNFP17g03b	–	1280	–	**700**	–	5200
CCNFP17g03c	–	1560	–	**1100**	–	7120
CCNFP17g04a	–	1840	–	**900**	–	8280
CCNFP17g04b	–	1920	–	**1100**	–	4880
CCNFP17g04c	–	2080	–	**1900**	–	8000
CCNFP17g05a	–	2400	–	**1400**	–	9720
CCNFP17g05b	–	1400	–	**700**	–	6360
CCNFP17g05c	–	3400	–	**1100**	–	6720
CCNFP17g06a	–	14760	–	**8000**	–	14320
CCNFP17g06b	–	12120	–	**11300**	2.95e-6	–
CCNFP17g06c	–	**4040**	–	16400	–	16400
CCNFP17g07a	–	1440	–	**600**	–	5160
CCNFP17g07b	–	16480	–	**2400**	2.47e-5	–
CCNFP17g07c	–	1480	–	**1100**	–	5560
CCNFP17g08a	–	1200	–	**600**	–	5040
CCNFP17g08b	–	1120	–	**300**	–	1920
CCNFP17g08c	–	1760	–	**900**	–	5800
CCNFP17g09a	–	1280	–	**600**	–	4760
CCNFP17g09b	–	1360	–	**500**	–	5240
CCNFP17g09c	–	1160	–	**500**	–	3360
CCNFP17g10a	–	1920	–	**1700**	–	6200
CCNFP17g10b	–	2000	–	**900**	–	4720
CCNFP17g10c	–	1400	–	**800**	–	5080
CCNFP19g01a	–	4440	–	**2800**	–	6560
CCNFP19g01b	–	**1600**	–	13900	–	6480
CCNFP19g01c	–	20760	–	**8000**	2.38e-4	–
CCNFP19g02a	–	2400	–	**1400**	–	10200
CCNFP19g02b	–	**1360**	–	2000	–	5280
CCNFP19g02c	–	1400	–	**700**	–	6560
CCNFP19g03a	–	4520	–	**1000**	–	7720
CCNFP19g03b	–	1320	–	**600**	–	5880
CCNFP19g03c	–	13480	–	**4600**	–	9320
CCNFP19g04a	–	4240	–	**1800**	–	10720
CCNFP19g04b	–	1240	–	**800**	–	4440
CCNFP19g04c	–	1680	–	**1100**	–	7560
CCNFP19g05a	–	3960	–	**1500**	–	4520
CCNFP19g05b	–	520	–	**300**	–	1360
CCNFP19g05c	–	9240	–	**1100**	–	5440

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP19g06a	–	2360	–	**1700**	–	11840
CCNFP19g06b	–	**6680**	–	9600	–	14440
CCNFP19g06c	3.30e-5	–	–	**5000**	–	9200
CCNFP19g07a	–	1560	–	**800**	–	6960
CCNFP19g07b	–	**2080**	–	3100	–	12320
CCNFP19g07c	–	5840	–	**1500**	–	10960
CCNFP19g08a	–	2240	–	**1400**	–	7560
CCNFP19g08b	–	1400	–	**900**	–	5520
CCNFP19g08c	–	1480	–	**800**	–	5600
CCNFP19g09a	–	7040	–	**1300**	–	10760
CCNFP19g09b	–	1760	–	**1200**	–	7000
CCNFP19g09c	–	2520	–	**1200**	–	5000
CCNFP19g10a	–	3080	–	**1800**	–	7360
CCNFP19g10b	–	2720	–	**1200**	–	7440
CCNFP19g10c	–	2640	–	**1300**	–	7120
CCNFP25g01a	–	5280	–	**3500**	–	17280
CCNFP25g01b	7.19e-5	**–**	5.60e-4	–	5.03e-4	–
CCNFP25g01c	4.01e-5	–	–	**11600**	1.85e-3	–
CCNFP25g02a	–	3160	–	**2800**	–	17680
CCNFP25g02b	–	2240	–	**1800**	–	11200
CCNFP25g02c	–	1800	–	**1100**	–	9640
CCNFP25g03a	–	15120	–	**5400**	3.07e-5	–
CCNFP25g03b	–	1960	–	**1400**	–	10040
CCNFP25g03c	–	1480	–	**1200**	–	7240
CCNFP25g04a	–	11360	–	**2300**	–	10160
CCNFP25g04b	–	2000	–	**800**	–	5720
CCNFP25g04c	–	**2560**	–	3600	–	13440
CCNFP25g05a	–	1840	–	**900**	–	6080
CCNFP25g05b	–	1480	–	**900**	–	6240
CCNFP25g05c	–	1480	–	**700**	–	7240
CCNFP25g06a	–	21600	–	**5800**	–	9280
CCNFP25g06b	2.53e-5	–	–	**3900**	2.53e-5	–
CCNFP25g06c	5.37e-5	–	–	19800	–	**16960**
CCNFP25g07a	–	**3280**	–	9400	–	12680
CCNFP25g07b	–	**3880**	–	4200	–	9760
CCNFP25g07c	–	10240	–	**5600**	–	8400
CCNFP25g08a	–	1560	–	**1300**	–	9640
CCNFP25g08b	–	1920	–	**1100**	–	7640
CCNFP25g08c	–	4080	–	**2600**	–	16400
CCNFP25g09a	–	12600	–	**1700**	–	9440
CCNFP25g09b	–	2960	–	**1200**	–	6360

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP25g09c	–	2120	–	**1100**	–	6760
CCNFP25g10a	–	2920	–	**1300**	–	9480
CCNFP25g10b	–	2680	–	**1600**	–	10800
CCNFP25g10c	–	3440	–	**1700**	–	13240
CCNFP30g01a	–	11560	–	**5100**	4.66e-4	–
CCNFP30g01b	–	**6040**	–	11400	–	16520
CCNFP30g01c	4.20e-5	–	–	**17500**	9.42e-5	–
CCNFP30g02a	–	2200	–	**1500**	–	10320
CCNFP30g02b	–	2080	–	**1300**	–	9400
CCNFP30g02c	–	5600	–	**4100**	–	18280
CCNFP30g03a	–	**5440**	–	17000	3.20e-4	–
CCNFP30g03b	–	8520	–	**4100**	–	12320
CCNFP30g03c	–	1840	–	**1000**	–	7880
CCNFP30g04a	–	15680	–	**2400**	–	9760
CCNFP30g04b	–	5160	–	**2300**	–	14640
CCNFP30g04c	–	3160	–	**2500**	–	12760
CCNFP30g05a	–	1920	–	**1200**	–	8160
CCNFP30g05b	–	2600	–	**1700**	–	13520
CCNFP30g05c	–	4040	–	**2200**	–	15000
CCNFP30g06a	–	3040	–	**1600**	–	13440
CCNFP30g06b	–	2880	–	**2100**	–	19080
CCNFP30g06c	1.95e-3	–	7.82e-4	**–**	2.17e-3	–
CCNFP30g07a	–	4640	–	**1200**	–	9000
CCNFP30g07b	5.55e-6	–	–	**4000**	2.12e-4	–
CCNFP30g07c	–	8400	–	**1400**	–	10440
CCNFP30g08a	–	2200	–	**1700**	–	9520
CCNFP30g08b	–	1680	–	**1000**	–	7560
CCNFP30g08c	–	1880	–	**900**	–	8280
CCNFP30g09a	–	3520	–	**1600**	–	9080
CCNFP30g09b	–	12640	–	**3700**	–	13560
CCNFP30g09c	–	8600	–	**3200**	–	10680
CCNFP30g10a	–	5040	–	**2300**	–	15280
CCNFP30g10b	–	2480	–	**1300**	–	7920
CCNFP30g10c	–	3000	–	**1900**	–	12400
CCNFP40g01a	–	12400	–	**6000**	1.30e-4	–
CCNFP40g01b	3.77e-3	**–**	4.06e-3	–	1.32e-2	–
CCNFP40g01c	–	10040	–	**6400**	1.20e-4	–
CCNFP40g02a	8.07e-5	–	–	**17200**	1.52e-4	–
CCNFP40g02b	–	8880	–	**4400**	–	21240
CCNFP40g02c	5.16e-5	–	–	**9800**	2.15e-4	–
CCNFP40g03a	–	2560	–	**2000**	–	14800

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
CCNFP40g03b	–	8560	–	**3300**	–	21000
CCNFP40g03c	–	**2640**	–	5300	–	15000
CCNFP40g04a	–	2400	–	**2000**	–	13440
CCNFP40g04b	4.08e-5	–	–	**6500**	1.45e-4	–
CCNFP40g04c	–	1960	–	**1300**	–	10840
CCNFP40g05a	–	3960	–	**3600**	–	12600
CCNFP40g05b	–	2400	–	**1500**	–	13600
CCNFP40g05c	–	2440	–	**1700**	–	12160
CCNFP50g01a	–	12600	–	**5200**	–	21520
CCNFP50g01b	1.02e-3	–	1.36e-4	**–**	3.13e-3	–
CCNFP50g01c	2.00e-4	**–**	2.25e-4	–	5.70e-3	–
CCNFP50g02a	–	20240	–	**11300**	2.60e-5	–
CCNFP50g02b	–	18360	–	**6300**	5.11e-4	–
CCNFP50g02c	–	**1880**	–	2600	–	10640
CCNFP50g03a	–	2920	–	**1500**	–	15120
CCNFP50g03b	–	1400	–	**700**	–	8600
CCNFP50g03c	–	2520	–	**1800**	–	14800
CCNFP50g04a	–	1840	–	**1600**	–	12760
CCNFP50g04b	1.25e-5	–	–	**3000**	3.45e-5	–
CCNFP50g04c	–	4200	–	**2400**	–	16560
CCNFP50g05a	–	22560	–	**1700**	–	13200
CCNFP50g05b	–	3080	–	**1400**	–	10280
CCNFP50g05c	–	8080	–	**4100**	–	17840
SmallWorldN40g01a	–	22240	–	**8800**	2.45e-4	–
SmallWorldN40g01b	–	**2080**	–	2100	–	15320
SmallWorldN40g01c	–	18280	–	**7000**	–	14720
SmallWorldN40g02a	–	2920	–	**1500**	–	14440
SmallWorldN40g02b	–	2520	–	**1800**	–	16400
SmallWorldN40g02c	–	3240	–	**1800**	–	13200
SmallWorldN40g03a	–	3240	–	**1700**	–	14600
SmallWorldN40g03b	–	3280	–	**2200**	–	17320
SmallWorldN40g03c	–	3960	–	**2200**	–	13440
SmallWorldN40g04a	–	3520	–	**2200**	–	16400
SmallWorldN40g04b	–	4000	–	**2000**	–	15000
SmallWorldN40g04c	–	2600	–	**1400**	–	10960
SmallWorldN40g05a	–	4720	–	**2100**	–	16200
SmallWorldN40g05b	–	2640	–	**1700**	–	12760
SmallWorldN40g05c	–	2560	–	**1500**	–	13960
SmallWorldN50g01a	–	13200	–	**6400**	1.54e-4	–
SmallWorldN50g01b	2.83e-5	–	–	**5300**	8.89e-5	–
SmallWorldN50g01c	–	5840	–	**5300**	6.85e-5	–

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
SmallWorldN50g02a	–	4040	–	**2900**	–	18640
SmallWorldN50g02b	–	2840	–	**1400**	–	14720
SmallWorldN50g02c	–	3360	–	**2500**	–	16720
SmallWorldN50g03a	–	6720	–	**3000**	–	23000
SmallWorldN50g03b	–	6800	–	**2000**	–	19760
SmallWorldN50g03c	–	3240	–	**1700**	–	13800
SmallWorldN50g04a	–	3600	–	**2300**	–	16520
SmallWorldN50g04b	–	5560	–	**2700**	–	18160
SmallWorldN50g04c	–	8440	–	**4900**	–	21880
SmallWorldN50g05a	–	8400	–	**4000**	–	22720
SmallWorldN50g05b	–	7920	–	**5200**	–	22600
SmallWorldN50g05c	–	2960	–	**1600**	–	15400
SmallWorldN75g01a	5.16e-5	–	3.23e-5	**–**	1.64e-3	–
SmallWorldN75g01b	–	**9480**	–	11700	3.35e-3	–
SmallWorldN75g01c	–	**2800**	–	6400	1.13e-4	–
SmallWorldN75g02a	–	5040	–	**3900**	1.71e-5	–
SmallWorldN75g02b	–	20240	–	**5400**	1.53e-4	–
SmallWorldN75g02c	–	8640	–	**4900**	5.04e-5	–
SmallWorldN75g03a	–	17080	–	**6700**	2.64e-4	–
SmallWorldN75g03b	–	5200	–	**3700**	2.07e-5	–
SmallWorldN75g03c	–	4280	–	**3500**	–	21400
SmallWorldN75g04a	–	7120	–	**4000**	1.07e-5	–
SmallWorldN75g04b	–	5800	–	**5300**	6.24e-5	–
SmallWorldN75g04c	–	8080	–	**3900**	1.19e-5	–
SmallWorldN75g05a	–	6240	–	**4200**	2.45e-6	–
SmallWorldN75g05b	–	5240	–	**4800**	–	22800
SmallWorldN75g05c	–	5200	–	**4100**	1.49e-5	–
SmallWorldN100g1a	2.25e-5	**–**	5.28e-5	–	5.15e-3	–
SmallWorldN100g1b	7.79e-5	–	–	**14800**	4.24e-3	–
SmallWorldN100g1c	3.19e-5	**–**	7.89e-5	–	7.99e-3	–
SmallWorldN100g2a	–	6400	–	**5300**	4.71e-4	–
SmallWorldN100g2b	–	4600	–	**3200**	9.89e-6	–
SmallWorldN100g2c	1.51e-4	–	1.11e-4	**–**	1.77e-3	–
SmallWorldN100g3a	–	4600	–	**3800**	4.49e-5	–
SmallWorldN100g3b	–	11800	–	**9400**	1.13e-3	–
SmallWorldN100g3c	–	11440	–	**8100**	1.04e-3	–
SmallWorldN100g4a	–	18640	–	**6300**	1.01e-3	–
SmallWorldN100g4b	–	7480	–	**4900**	2.08e-4	–
SmallWorldN100g4c	–	14680	–	**10000**	9.20e-4	–
SmallWorldN100g5a	–	10840	–	**6500**	6.54e-5	–
SmallWorldN100g5b	–	19840	–	**11200**	9.45e-4	–

Continued on next page

Graph	**Threshold-ACO**		**Reinforcement-ACO**		**Max-Min Ant System**	
	Avg.Dev	*Sol*	*Avg.Dev*	*Sol*	*Avg.Dev*	*Sol*
SmallWorldN100g5c	–	11280	–	**4800**	3.87e-4	–

Appendix B

Stagnation Experiment Results

Table B.1: A comparison of how well the stagnation avoidance strategies coped with MCFP instances where at least one stagnated. The table list only the graphs where at least one algorithm stagnated. The first column *Avg.Dev* list the deviation between the average solution and the global minimum. A value of 0.5 imply that the average discovered solution cost was 1.5 times the minimum cost. In other words, closer to 0 is better. The second column list the number of runs that stagnated before discovering the global optimum out of 100. Every instance was solved 100 times with optimal parameter configuration. The algorithms were allowed to generate at most 24000 solutions. A '−' in the *Avg.Dev* column indicate that the algorithm always converged to the global optimum.

Graph	**FT-ACO**		**SR-ACO**		**MMAS**		$\text{AS}_{\boldsymbol{rank}}$	
	Avg.Dev	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*
CCNFP10g05a	−	**0**	−	**0**	−	**0**	3.43e-4	5
CCNFP10g07c	−	**0**	−	**0**	−	**0**	4.72e-4	3
CCNFP12g01b	−	**0**	−	**0**	−	**0**	3.74e-4	1
CCNFP12g01c	1.49e-4	19	−	**0**	1.48e-5	4	3.57e-4	54
CCNFP12g03b	−	**0**	−	**0**	−	**0**	6.17e-5	3
CCNFP12g04c	−	**0**	−	**0**	−	**0**	3.21e-5	1
CCNFP12g06a	−	**0**	−	**0**	6.52e-3	16	2.06e-2	51
CCNFP12g06b	−	**0**	−	**0**	−	**0**	2.80e-3	41
CCNFP12g07b	−	**0**	−	**0**	−	**0**	1.08e-4	29
CCNFP12g08a	−	**0**	−	**0**	−	**0**	1.19e-4	2
CCNFP15g01a	1.62e-3	10	−	**0**	8.06e-3	39	1.87e-2	85

Continued on next page

Graph	**FT-ACO**		**SR-ACO**		**MMAS**		AS$_{rank}$	
	Avg.Dev	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*
CCNFP15g01c	1.24e-3	3	–	**0**	–	**0**	3.16e-2	75
CCNFP15g06a	–	**0**	–	**0**	–	**0**	1.09e-4	1
CCNFP15g06b	–	**0**	–	**0**	–	**0**	1.47e-3	34
CCNFP15g06c	–	**0**	–	**0**	–	**0**	3.44e-3	34
CCNFP15g07a	–	**0**	–	**0**	1.37e-3	5	5.34e-3	19
CCNFP15g08a	–	**0**	–	**0**	–	**0**	1.68e-4	5
CCNFP17g01a	1.05e-3	22	9.59e-5	2	–	**0**	3.34e-3	46
CCNFP17g01b	1.57e-6	4	–	**0**	1.93e-4	9	5.47e-3	75
CCNFP17g06a	–	**0**	–	**0**	–	**0**	3.57e-3	40
CCNFP17g06b	–	**0**	–	**0**	2.95e-6	1	3.84e-5	13
CCNFP17g06c	–	**0**	–	**0**	–	**0**	1.46e-2	62
CCNFP17g07b	–	**0**	–	**0**	2.47e-5	7	1.27e-4	36
CCNFP19g01a	–	**0**	–	**0**	–	**0**	5.72e-4	2
CCNFP19g01c	–	**0**	–	**0**	2.38e-4	4	7.07e-3	65
CCNFP19g02a	–	**0**	–	**0**	–	**0**	2.86e-4	3
CCNFP19g03c	–	**0**	–	**0**	–	**0**	8.78e-5	4
CCNFP19g04a	–	**0**	–	**0**	–	**0**	2.35e-4	4
CCNFP19g06b	–	**0**	–	**0**	–	**0**	2.47e-3	33
CCNFP19g06c	3.30e-5	1	–	**0**	–	**0**	8.36e-4	12
CCNFP19g07b	–	**0**	–	**0**	–	**0**	4.82e-4	4
CCNFP19g07c	–	**0**	–	**0**	–	**0**	2.86e-4	3
CCNFP19g09a	–	**0**	–	**0**	–	**0**	6.51e-5	9
CCNFP25g01a	–	**0**	–	**0**	–	**0**	9.05e-4	41
CCNFP25g01b	7.19e-5	1	5.60e-4	9	5.03e-4	7	5.43e-3	67
CCNFP25g01c	4.01e-5	1	–	**0**	1.85e-3	46	4.00e-3	76
CCNFP25g02a	–	**0**	–	**0**	–	**0**	2.07e-3	39
CCNFP25g02b	–	**0**	–	**0**	–	**0**	5.18e-5	3
CCNFP25g03a	–	**0**	–	**0**	3.07e-5	1	1.62e-3	37
CCNFP25g04a	–	**0**	–	**0**	–	**0**	5.31e-5	1
CCNFP25g04c	–	**0**	–	**0**	–	**0**	1.18e-4	3
CCNFP25g06a	–	**0**	–	**0**	–	**0**	3.98e-5	4
CCNFP25g06b	2.53e-5	1	–	**0**	2.53e-5	1	1.69e-3	50
CCNFP25g06c	5.37e-5	2	–	**0**	–	**0**	1.89e-3	61
CCNFP25g07a	–	**0**	–	**0**	–	**0**	2.17e-4	8
CCNFP25g07b	–	**0**	–	**0**	–	**0**	1.45e-3	20
CCNFP25g07c	–	**0**	–	**0**	–	**0**	3.30e-4	4
CCNFP25g08c	–	**0**	–	**0**	–	**0**	6.75e-4	35
CCNFP25g10c	–	**0**	–	**0**	–	**0**	2.36e-6	1
CCNFP30g01a	–	**0**	–	**0**	4.66e-4	2	1.06e-2	42
CCNFP30g01b	–	**0**	–	**0**	–	**0**	1.97e-3	42
CCNFP30g01c	4.20e-5	13	–	**0**	9.42e-5	21	1.48e-3	77

Continued on next page

Graph	**FT-ACO**		**SR-ACO**		**MMAS**		AS$_{rank}$	
	Avg.Dev	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*
CCNFP30g02a	–	**0**	–	**0**	–	**0**	1.95e-4	2
CCNFP30g02c	–	**0**	–	**0**	–	**0**	1.72e-3	30
CCNFP30g03a	–	**0**	–	**0**	3.20e-4	5	1.36e-3	18
CCNFP30g04a	–	**0**	–	**0**	–	**0**	9.90e-5	2
CCNFP30g04b	–	**0**	–	**0**	–	**0**	1.38e-5	2
CCNFP30g04c	–	**0**	–	**0**	–	**0**	9.74e-6	2
CCNFP30g05c	–	**0**	–	**0**	–	**0**	8.15e-6	1
CCNFP30g06a	–	**0**	–	**0**	–	**0**	2.14e-3	19
CCNFP30g06b	–	**0**	–	**0**	–	**0**	2.56e-3	34
CCNFP30g06c	1.95e-3	10	7.82e-4	4	2.17e-3	20	1.45e-2	90
CCNFP30g07a	–	**0**	–	**0**	–	**0**	3.43e-4	2
CCNFP30g07b	5.55e-6	3	–	**0**	2.12e-4	5	3.67e-3	68
CCNFP30g07c	–	**0**	–	**0**	–	**0**	9.28e-5	6
CCNFP30g09b	–	**0**	–	**0**	–	**0**	3.13e-4	20
CCNFP30g09c	–	**0**	–	**0**	–	**0**	9.99e-5	2
CCNFP30g10a	–	**0**	–	**0**	–	**0**	6.11e-5	4
CCNFP40g01a	–	**0**	–	**0**	1.30e-4	7	6.76e-3	86
CCNFP40g01b	3.77e-3	89	4.06e-3	81	1.32e-2	100	2.27e-2	100
CCNFP40g01c	–	**0**	–	**0**	1.20e-4	6	6.96e-3	87
CCNFP40g02a	8.07e-5	2	–	**0**	1.52e-4	2	4.81e-3	66
CCNFP40g02b	–	**0**	–	**0**	–	**0**	1.46e-3	59
CCNFP40g02c	5.16e-5	13	–	**0**	2.15e-4	79	1.75e-3	94
CCNFP40g03a	–	**0**	–	**0**	–	**0**	6.90e-5	11
CCNFP40g03b	–	**0**	–	**0**	–	**0**	3.20e-4	42
CCNFP40g03c	–	**0**	–	**0**	–	**0**	3.18e-4	9
CCNFP40g04b	4.08e-5	9	–	**0**	1.45e-4	32	5.84e-4	59
CCNFP40g05a	–	**0**	–	**0**	–	**0**	6.70e-6	1
CCNFP40g05b	–	**0**	–	**0**	–	**0**	1.83e-5	3
CCNFP50g01a	–	**0**	–	**0**	–	**0**	7.51e-3	71
CCNFP50g01b	1.02e-3	30	1.36e-4	4	3.13e-3	88	1.02e-2	100
CCNFP50g01c	2.00e-4	8	2.25e-4	9	5.70e-3	98	2.44e-2	100
CCNFP50g02a	–	**0**	–	**0**	2.60e-5	2	1.37e-3	85
CCNFP50g02b	–	**0**	–	**0**	5.11e-4	47	4.97e-3	96
CCNFP50g02c	–	**0**	–	**0**	–	**0**	5.92e-4	5
CCNFP50g03a	–	**0**	–	**0**	–	**0**	8.44e-5	9
CCNFP50g03c	–	**0**	–	**0**	–	**0**	2.34e-4	15
CCNFP50g04a	–	**0**	–	**0**	–	**0**	1.09e-5	2
CCNFP50g04b	1.25e-5	4	–	**0**	3.45e-5	11	2.24e-4	48
CCNFP50g05a	–	**0**	–	**0**	–	**0**	3.25e-4	8
CCNFP50g05c	–	**0**	–	**0**	–	**0**	1.27e-4	13
SmallWorldN40g01a	–	**0**	–	**0**	2.45e-4	**0**	7.97e-3	89

Continued on next page

Graph	**FT-ACO**		**SR-ACO**		**MMAS**		AS$_{rank}$	
	Avg.Dev	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*
SmallWorldN40g01b	–	**0**	–	**0**	–	**0**	2.87e-4	11
SmallWorldN40g01c	–	**0**	–	**0**	–	**0**	1.16e-3	32
SmallWorldN40g02a	–	**0**	–	**0**	–	**0**	2.40e-5	4
SmallWorldN40g02b	–	**0**	–	**0**	–	**0**	3.11e-5	4
SmallWorldN40g02c	–	**0**	–	**0**	–	**0**	1.35e-5	2
SmallWorldN40g03a	–	**0**	–	**0**	–	**0**	1.22e-4	4
SmallWorldN40g03b	–	**0**	–	**0**	–	**0**	2.15e-4	9
SmallWorldN40g03c	–	**0**	–	**0**	–	**0**	1.37e-5	4
SmallWorldN40g04a	–	**0**	–	**0**	–	**0**	5.01e-5	7
SmallWorldN40g04b	–	**0**	–	**0**	–	**0**	5.80e-5	5
SmallWorldN40g05a	–	**0**	–	**0**	–	**0**	5.31e-5	6
SmallWorldN40g05b	–	**0**	–	**0**	–	**0**	5.00e-6	1
SmallWorldN40g05c	–	**0**	–	**0**	–	**0**	1.18e-5	1
SmallWorldN50g01a	–	**0**	–	**0**	1.54e-4	**0**	5.68e-3	78
SmallWorldN50g01b	2.83e-5	1	–	**0**	8.89e-5	3	2.59e-3	60
SmallWorldN50g01c	–	**0**	–	**0**	6.85e-5	1	1.11e-2	77
SmallWorldN50g02a	–	**0**	–	**0**	–	**0**	3.90e-4	20
SmallWorldN50g02b	–	**0**	–	**0**	–	**0**	1.19e-4	7
SmallWorldN50g02c	–	**0**	–	**0**	–	**0**	6.45e-5	13
SmallWorldN50g03a	–	**0**	–	**0**	–	**0**	6.09e-4	44
SmallWorldN50g03b	–	**0**	–	**0**	–	**0**	1.22e-4	27
SmallWorldN50g03c	–	**0**	–	**0**	–	**0**	7.73e-5	3
SmallWorldN50g04a	–	**0**	–	**0**	–	**0**	1.26e-4	10
SmallWorldN50g04b	–	**0**	–	**0**	–	**0**	1.81e-4	16
SmallWorldN50g04c	–	**0**	–	**0**	–	**0**	4.48e-4	47
SmallWorldN50g05a	–	**0**	–	**0**	–	**0**	5.68e-4	49
SmallWorldN50g05b	–	**0**	–	**0**	–	**0**	5.96e-4	45
SmallWorldN50g05c	–	**0**	–	**0**	–	**0**	8.17e-5	8
SmallWorldN75g01a	5.16e-5	2	3.23e-5	5	1.64e-3	**0**	8.31e-3	99
SmallWorldN75g01b	–	**0**	–	**0**	3.35e-3	**0**	1.12e-2	100
SmallWorldN75g01c	–	**0**	–	**0**	1.13e-4	9	7.43e-3	84
SmallWorldN75g02a	–	**0**	–	**0**	1.71e-5	3	1.86e-3	73
SmallWorldN75g02b	–	**0**	–	**0**	1.53e-4	**0**	3.17e-3	95
SmallWorldN75g02c	–	**0**	–	**0**	5.04e-5	4	2.94e-3	78
SmallWorldN75g03a	–	**0**	–	**0**	2.64e-4	**0**	3.36e-3	95
SmallWorldN75g03b	–	**0**	–	**0**	2.07e-5	6	2.71e-3	78
SmallWorldN75g03c	–	**0**	–	**0**	–	**0**	6.29e-4	50
SmallWorldN75g04a	–	**0**	–	**0**	1.07e-5	**0**	1.40e-3	81
SmallWorldN75g04b	–	**0**	–	**0**	6.24e-5	36	1.12e-3	90
SmallWorldN75g04c	–	**0**	–	**0**	1.19e-5	5	1.29e-3	78
SmallWorldN75g05a	–	**0**	–	**0**	2.45e-6	3	5.73e-4	67

Continued on next page

Graph	**FT-ACO**		**SR-ACO**		**MMAS**		AS$_{rank}$	
	Avg.Dev	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*	*Avg.Dev*	*Stg*
SmallWorldN75g05b	–	**0**	–	**0**	–	**0**	1.02e-3	63
SmallWorldN75g05c	–	**0**	–	**0**	1.49e-5	**0**	2.10e-3	84
SmallWorldN100g1a	2.25e-5	3	5.28e-5	2	5.15e-3	**0**	1.69e-2	100
SmallWorldN100g1b	7.79e-5	15	–	**0**	4.24e-3	**0**	1.38e-2	100
SmallWorldN100g1c	3.19e-5	31	7.89e-5	10	7.99e-3	**0**	1.95e-2	100
SmallWorldN100g2a	–	**0**	–	**0**	4.71e-4	78	3.09e-3	95
SmallWorldN100g2b	–	**0**	–	**0**	9.89e-6	9	1.96e-3	81
SmallWorldN100g2c	1.51e-4	17	1.11e-4	9	1.77e-3	**0**	6.62e-3	100
SmallWorldN100g3a	–	**0**	–	**0**	4.49e-5	24	1.87e-3	86
SmallWorldN100g3b	–	**0**	–	**0**	1.13e-3	**0**	7.19e-3	98
SmallWorldN100g3c	–	**0**	–	**0**	1.04e-3	**0**	5.30e-3	98
SmallWorldN100g4a	–	**0**	–	**0**	1.01e-3	**0**	5.24e-3	99
SmallWorldN100g4b	–	**0**	–	**0**	2.08e-4	60	3.17e-3	97
SmallWorldN100g4c	–	**0**	–	**0**	9.20e-4	**0**	4.24e-3	99
SmallWorldN100g5a	–	**0**	–	**0**	6.54e-5	44	2.46e-3	83
SmallWorldN100g5b	–	**0**	–	**0**	9.45e-4	**0**	4.47e-3	100
SmallWorldN100g5c	–	**0**	–	**0**	3.87e-4	**0**	3.20e-3	96

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* 1st ed. Upper Saddle River, New Jersey: Prentice Hall, Feb. 1993.

[2] F. Altiparmak and I. Karaoglan. "A genetic ant colony optimization approach for concave cost transportation problems". In: *2007 IEEE Congress on Evolutionary Computation.* IEEE, Sept. 2007, pp. 1685–1692.

[3] T. C. Baker, SE Van Vorhis Key, and L. K. Gaston. "Bait-preference Tests for the Argentine Ant (Hymenoptera : Formicidae)". In: *Journal of Economic Entomology* 78.5 (1985), pp. 1083–1088.

[4] John E. Beasley. *Network Flow - Single commodity - Concave Costs - Single Source Uncapacitated.* URL: `http://people.brunel.ac.uk/%7B~%7Dmastjjb/jeb/orlib/netflowccinfo.html` (visited on 05/05/2016).

[5] Samuel N Beshers and Jennifer H Fewell. "Models of division of labor in social insects." In: *Annual review of entomology* 46 (2001), pp. 413–440.

[6] Christian Blum and Maria J. Blesa. "New metaheuristic approaches for the edge-weighted k-cardinality tree problem". In: *Computers and Operations Research* 32.6 (2005), pp. 1355–1377.

[7] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. "Fixed Response Thresholds and the Regulation of Division of Labor in Insect Societies". In: *Bulletin of Mathematical Biology* 60 (1998), pp. 753–807.

[8] Eric Bonabeau, Guy Theraulaz, and Jean-Louis Deneubourg. "Quantitative study of the fixed threshold model for the regulation of division of labot in insect societies." In: *The Royal Society* Proc R Soc.263 (1996), pp. 1565–1569.

[9] Eric Bonabeau et al. “Adaptive Task Allocation Inspired by a Model of Division of Labor in Social Insects”. In: *Biocomputing and Emergent Computation* 8 (1997), pp. 36–45.

[10] Thang N. Bui and Catherine M. Zrncic. “An ant-based algorithm for finding degree-constrained minimum spanning tree”. In: *Gecco* (2006), p. 11.

[11] Bernd Bullnheimer, Richard F Hartl, and Christine Strauss. “A New Rank Based Version of the Ant System — A Computational Study”. In: *Central European Journal for Operations Research and Economics* 7 (1997), pp. 25–38.

[12] Chia H Chen and Ching Jung Ting. “Combining Lagrangian heuristic and Ant Colony System to solve the Single Source Capacitated Facility Location Problem”. In: *Transportation Research Part E: Logistics and Transportation Review* 44.6 (2008), pp. 1099–1122.

[13] Broderick Crawford and Carlos Castro. “Integrating Lookahead and Post Processing Procedures with ACO for Solving Set Partitioning and Covering Problems”. In: *Artificial Intelligence and Soft Computing – ICAISC 2006.* Springer Berlin Heidelberg, 2006, pp. 1082–1090.

[14] Cl Detrain and J. M. Pasteels. “Caste differences in behavioral thresholds as a basis for polyethism during food recruitment in the ant, Pheidole pallidula (Nyl.) (Hymenoptera: Myrmicinae)”. In: *Journal of Insect Behavior* 4.2 (1991), pp. 157–176.

[15] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. “Ant algorithms and stigmergy”. In: *Future Generation Computer Systems* 16.8 (2000), pp. 851–871.

[16] Marco Dorigo and Gianni Di Caro. “Ant colony optimization: a new meta-heuristic”. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).* Vol. 2. IEEE, 1999, pp. 1470–1477.

[17] Marco Dorigo and Luca Maria Gambardella. “Ant colony system: A cooperative learning approach to the traveling salesman problem”. In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 53–66.

[18] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. “Ant system: Optimization by a colony of cooperating agents”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 26.1 (1996), pp. 29–41.

[19] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization.* MIT Press, 2004.

[20] Marco Dorigo and Thomas Stützle. "The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances". In: *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003, pp. 250–285.

[21] Walid Elloumi et al. "A comparative study of the improvement of performance using a PSO modified by ACO applied to TSP". In: *Applied Soft Computing Journal* 25 (2014), pp. 234–241.

[22] Jennifer H. Fewell and Susan M. Bertram. "Division of labor in a dynamic environment: Response by honeybees (Apis mellifera) to graded changes in colony pollen stores". In: *Behavioral Ecology and Sociobiology* 46.3 (1999), pp. 171–179.

[23] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. "The biological principles of swarm intelligence". In: *Swarm Intelligence* 1.May (2007), pp. 3–31.

[24] G M Guisewite and Panos M Pardalos. "Minimum concave-cost network flow problems: Applications, complexity, and algorithms". In: *Annals of Operations Research* 25.1 (1990), pp. 75–99.

[25] Petter Holme and Beom Jun Kim. "Growing scale-free networks with tunable clustering." In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 65.2 Pt 2 (Feb. 2002), p. 026107.

[26] Yun Kang and Guy Theraulaz. "Dynamical models of task organization in social insect colonies". In: *arXiv preprint arXiv:1511.04769* (2015), pp. 1–34.

[27] Jeff Kennington and Ed Unger. "A New Branch-and-Bound Algorithm for the Fixed-Charge Transportation Problem". In: *Management Science* 22.10 (1976), pp. 1116–1126.

[28] Bruce W. Lamar. *Network Optimization Problems: Algorithms, Applications and Complexity*. Ed. by D.Z. Du and P.M. Pardalos. World Scientific, 1991, pp. 147–167.

[29] Tianjun Liao et al. "An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization". In: *Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference* (2011), pp. 125–132.

[30] Ke Ma et al. "Power law and small world properties in a comparison of traffic city networks". In: *Chinese Science Bulletin* 56.34 (Dec. 2011), pp. 3731–3735.

[31] Laurens Van Der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[32] Marta Sofia Rodrigues Monteiro, Dalila B M M Fontes, and Fernando A C C Fontes. “An Ant Colony Optimization Algorithm to Solve the Minimum Cost Network Flow Problem with Concave Cost Functions”. In: *Gecco-2011: Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference* (2011), pp. 139–145.

[33] Marta Sofia Rodrigues Monteiro, Dalila B M M Fontes, and Fernando A C C Fontes. “Concave Minimum Cost Network Flow Problems Solved with a Colony of Ants”. In: *Journal of Heuristics* 19.1 (2013), pp. 1–33.

[34] Mark EJ Newman. “The Structure and Function of Complex Networks”. In: *SIAM Review* 45.2 (Jan. 2003), pp. 167–256.

[35] Robert E Page and Sandra D Mitchell. “Self Organization and Adaptation in Insect Societies”. In: *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association* 1990.Volume Two: Symposia and Invited Papers (1990) (1990), pp. 289–298.

[36] Robert E Page and Sandra D Mitchell. “Self-organization and the evolution of division of labor”. In: *Apidologie* 29.1-2 (1998), pp. 171–190.

[37] Udatta S. Palekar, Mark H. Karwan, and Stanley Zionts. “A Branch-and-Bound Method for the Fixed Charge Transportation Problem”. In: *Management Science* 36.9 (Sept. 1990), pp. 1092–1105.

[38] Gene Ezia Robinson. “Regulation of division of labor in insect societies.” In: *Annual review of entomology* 37 (1992), pp. 637–665.

[39] Luís Santos, João Coutinho-Rodrigues, and John R Current. “An improved ant colony optimization based algorithm for the capacitated arc routing problem”. In: *Transportation Research Part B: Methodological* 44.2 (2010), pp. 246–266.

[40] T C Schneirla. “A unique case of circular milling in ants, considered in relation to trail folloing and the general problem of orientation”. In: *American Museum Novitates* 1253 (1944), pp. 1–26.

[41] Parongama Sen et al. “Small-world properties of the Indian railway network.” In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 67.3 Pt 2 (2003), p. 036106.

[42] T. Stützle and H. Hoos. “Improvements on the Ant-System: Introducing the MAX-MIN Ant System”. In: *Artificial Neural Nets and Genetic Algorithms.* Vienna: Springer Vienna, 1998, pp. 245–249.

[43] Thomas Stützle and Holger H Hoos. “Max-Min Ant System”. In: *Future Generation Computer Systems* 16.8 (2000), pp. 889–914.

[44] Guy Theraulaz, Eric Bonabeau, and Jean-louis Deneubourg. “Response threshold reinforcements and division of labour in insect societies”. In: *Proceedings of the Royal Society B: Biological Sciences* 265.1393 (1998), pp. 327–332.

[45] Harry Venables and Alfredo Moscardini. “An adaptive search heuristic for the capacitated fixed charge location problem”. In: *Ant Colony Optimization and Swarm ...* (2006), pp. 348–355.

[46] Harvey M. Wagner. “On a Class of Capacitated Transportation Problems”. In: *Management Science* 5.3 (Apr. 1959), pp. 304–318.

[47] Anja Weidenmüller. “The control of nest climate in bumblebee (Bombus terrestris) colonies: Interindividual variability and self reinforcement in fanning response”. In: *Behavioral Ecology* 15.1 (2004), pp. 120–128.

[48] Diana E Wheeler. “The Developmental Basis of Worker Caste Polymorphism in Ants”. In: *The American Naturalist* 138.5 (1991), pp. 1218–1238.

[49] Edward O Wilson. “The relation between caste ratios and division of labor in the ant genus Pheidole (Hymenoptera: Formicidae)”. In: *Behavioral Ecology and Sociobiology* 16.1 (Nov. 1984), pp. 89–98.

[50] Kuan Yew Wong and Phen Chiak See. “A new minimum pheromone threshold strategy (MPTS) for max-min ant system”. In: *Applied Soft Computing Journal* 9.3 (2009), pp. 882–888.

[51] Renbin Xiao. “Modeling and Simulation of Ant Colony’s Labor Division”. In: *Swarm Intelligence and Bio-Inspired Computation.* Elsevier, 2013, pp. 103–135.

[52] Zengwang Xu and Daniel Z. Sui. “Small-world characteristics on transportation networks: A perspective from network autocorrelation”. In: *Journal of Geographical Systems* 9.2 (2007), pp. 189–205.

[53] S Yan, Y L Shih, and C L Wang. “An ant colony system-based hybrid algorithm for square root concave cost transhipment problems”. In: *Engineering Optimization* 42.11 (2010), pp. 983–1001.

[54] Shangyao Yan et al. “Global and local search algorithms for concave cost transshipment problems”. In: *Journal of Global Optimization* 33.1 (2005), pp. 123–156.

[55] Willard I. Zangwill. “Minimum Concave Cost Flows in Certain Networks”. In: *Management Science* 14.7 (Mar. 1968), pp. 429–450.