DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2018

Ant Colony Optimization - Optimal Number of Ants

CHRISTOFFER LUNDELL JOHANSSON

LARS PETTERSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Ant Colony Optimization - Optimal Number of Ants

CHRISTOFFER LUNDELL JOHANSSON

LARS PETTERSSON

Bachelor's Thesis in Computer Science, DD142X
Date: June 6, 2018
Supervisor: Alexander Kozlov
Examiner: Örjan Ekeberg
Swedish title: Myrkoloni-optimering - Optimalt antal myror
KTH, School of Electrical Engineering and Computer Science

Abstract

The focus of this thesis paper is to study the impact the number of ants has on the found solution of the Ant Colony Optimization (ACO) metaheuristic when solving the Traveling Salesman Problem. The goal was to find out how the length of the computed tours change for different amounts of ants within a limited number of iterations. To study this, three well known versions of the ACO algorithm were implemented and tested: Min-Max Ant System (MMAS), Elitist Ant System (EliteAS) and Ranked Ant System (RankedAS).

The results showed trends that were consistent over several test cases. EliteAS and RankedAS which both utilize specialist ants showed clear signs that the number of specialists had a large influence on the length of solutions. Meanwhile, normal ants did not affect the solutions as much. MMAS and EliteAS only had a small variation on the answer, with lower amount of ants being more favorable. On the other hand, RankedAS performed better by a large margin when working with five specialists and a number of ants equaling the number of cities in the problem.

Sammanfattning

Målet med denna rapport var att studera hur antalet myror som används av Ant Colony Optimization (ACO) påverkar resultatet vid lösandet av Traveling Salesman Problem (TSP). Hur ändras lösningens längd med olika antal myror, när antalet iterationer som får användas är begränsat? För att få fram ett svar på frågan implementerades och testades tre välkända ACO algoritmer: Min-Max Ant System (MMAS), Elitist Ant System (EliteAS) och Ranked Ant System (RankedAS).

Efter implementering och utförlig testning så uppdagades trender som var konsistenta över flera testfall. För EliteAS och RankedAS, som båda förlitar sig på specialiserade myror, hade antalet specialister en stor påverkan på den funna längden. Normala myror hade istället en liten påverkan på slutresultatet. För MMAS och EliteAS så var skillnaden minimal, med en viss favör mot ett lägre antal myror. RankedAS hade en motsatt trend och hade bäst resultat med fem specialister och lika många normala myror som antalet städer i TSP instansen.

Contents

Chapter 1

Introduction

Humans have always been prone to curiosity, trying to understand the world around them and trying to find solutions to problems we are faced with. Every hurdle that we come across is examined, and in some way or another, overcome with a method specialized to tackle that kind of problem. Some problems are by their very nature harder than others to solve. Thus, even though some problems may seem fairly simple at a first glance, they usually show their true nature when it comes to finding the optimal solution. Problems of this kind are commonly solved using heuristics: approximation algorithms that find solutions to problems in more reasonable time spans.

The Ant Colony Optimization (ACO) metaheuristic has like several other algorithms taken inspiration from nature. Since it was first introduced, several branches of research have been dedicated to it and finding ways to solve complex problems using it. This research has inspired the creation of a number of books, workshops[1] and papers over the years with many new findings as the field is explored[7]. ACO was originally created to solve the Traveling Salesman Problem (TSP). But since its inception a number of specialized ACO versions have been developed, testing against other complex problems. One problem facing those implementing and using the algorithms in this family are the many and varied parameters that must be decided upon. Everything from weights of the randomness function, deterioration of pheromone trails, and number of ants have to be taken into consideration when trying to get the best possible solution.

All papers on the subject matter have their own ideas on how to best specialize and use these parameters, presented to the reader as pure data and numbers used. Particularly, the number of ants used is almost exclusively a constant number and is a subject that is rarely touched upon. Trying to find an answer to what number is best or why said number is used can be an exercise in frustration as no clear answer is presented. The papers on the ant system algorithms focus mostly on how to improve the algorithm itself, staving of stagnation with pheromones, as well as comparing it to other state of the art algorithms used today. Thus, this paper is instead dedicated to researching and testing several of these algorithms with a main focus on the ants themselves.

1.1 Problem statement

How does the number of ants and specialist ants impact the length of the computed tours? Solving for Traveling Salesman Problem using the Min-Max Ant System, Elitist Ant System and Ranked Ant System, with a limited set of iterations.

1.2 Scope

Three ant system algorithms were selected to be part of the study. The focus is to see how the ants and the number of them interact with the computed solutions. Thus, for all the algorithms used there will be no local optimization or other manipulation of the data. Only the raw output from the algorithms themselves are considered as to not pollute the output data and how it relates to the ants. All the algorithms differ in how the ants interact with one another and how they interact with the pheromones in order to get past the issue with stagnation. To get as much valid data as possible we implemented three different ant systems and tested all of them against the same three problem instances. To get a good idea on how to choose the amount of ants that best benefit the implementations used, the data was then compiled and compared for different amount of ants and between the different algorithms.

1.3 Purpose

There is so much research, time and effort put into the field of ACO that optimizing or adding to the field itself is beyond the scope of a bachelor thesis. Thus, the purpose of this thesis is focused on being academic and looking deeper into only a single aspect of the algorithms. When the use of ants is discussed in articles about ACO, the authors state that the amount of ants they used is what they found most optimal [6]. A reasoning of why a particular number is used is not discussed and is the main reason this thesis came about.

Chapter 2

Background

In this section we discuss the background and information needed to better understand the ACO algorithms. Furthermore, there is information on what the Traveling Salesman Problem is, why it is such a hard problem to solve, and how heuristics are used to find solutions to the problem.

2.1 Terminology

Following are the abbreviations and terminology used in this thesis, to aid further reading.

ACO: Ant Colony Optimization

EliteAS: Elitist Ant System

RankedAS: Ranked Ant System

MMAS: Min-Max Ant System

TSP: Traveling Salesman problem

2.2 NP-complete problems

There is a problem space within computer science that is called NP-complete, where NP stands for nondeterministic polynomial time. It

is a complexity class used to describe certain types of decision problems. Checking whether a solution is correct is fast and easy, taking polynomial time to make sure that the solution itself is valid. However, the task of solving the decision problems and finding a solution cannot be done in proper polynomial time. If a solution is found, there is no fast or defined way to know whether it is the optimal solution or not. Many instances of NP-complete problems require that all possible solutions be found before it is possible to make sure that a solution is optimal. As can be seen, a problem of this type will escalate in execution time even for a small data set. This becomes even more of a problem as the problem grows in size as it does not scale well. To further iterate on this problem, below we will discuss one of the most well known problems of this type.

2.3 Traveling Salesman Problem

TSP puts focus on a theoretical salesman. They have to visit all local cities in order to sell their products. After finishing with all sales in one city they move on to another not yet visited city and resume their work there. After visiting every single city, the only thing that remains is to return to the first city. The problem thus being: from a starting city, how do you construct a tour that visits every single city and returns to the starting city with the shortest possible traveling distance? It is easy and fast to validate that a given tour is a valid answer. However, it is here where the problem stated in section 2.2 manifests: How would one know whether this solution is the shortest one? Without comparing it to every other possible tour, there is no real way of knowing. This fact has inspired research that persists to this day. To solve this problem and get a useful answer within a reasonable amount of time, heuristics are used.

2.4 Heuristics

There are a number of ways to solve TSP. An easy way to do it is by simply traveling along a greedy path. That is, always choosing the city that is nearest the current city as the next part of the path, and doing

this repeatedly until the set of non-visited cities is exhausted, finally returning to the starting city. This is one of the easiest ways to describe what a heuristic is. Namely, an approximation of the optimal solution to a problem that is sufficiently good enough. As the NP-complete problem space does not have answers that can be computed in polynomial time, many strategies to get around this limitation have been engineered. Constructing a valid, greedy tour can be achieved relatively fast. Furthermore, applying local optimization and fine-tuning the tour will lead to a solution that is a good enough approximation of the best solution.

The general need for heuristics has necessitated a wide variety of metaheuristics to solve for these problems. Such heuristics are made in a way that allow for easy adaptability, meaning there is not much change needed when using them to solve many different problems. ACO is one of these metaheuristics. The final computed tour could be the shortest one, but it is not guaranteed. The goal is simply to get a decent solution in a quick manner. If it took days to calculate a near-optimal tour, it would not be of any practical use.

2.5 Ant colony optimization

The first time that ACO was brought forward and introduced to the community was in the PHD thesis "Optimization, Learning and Natural Algorithms" written by Marco Dorigo in 1992[3]. Within his paper were three algorithms with ants as their main source of inspiration. Over time, more and more interest was directed toward the ideas presented in the thesis. In modern times, only one of those algorithms still has time and effort put into it. As the two other algorithms fell to the sidelines, the remaining one became what is now known as ACO[9].

When ants forage for food, they spread out from the nest in order to find sustenance to bring back. Every individual ant leaves behind a small amount of pheromones on the path they take. The pheromones will attract other ants and subsequently, other ants will be more likely to take the same paths already traveled by past ants. When food is found they bring it back to the nest, or go back and forth between the nest and the closest food source. Every time an ant makes this trip the trail of pheromones is reinforced, making it stronger and more attrac-

tive, ensuring that even more ants join in the task of acquiring food from the source. The ants themselves cannot solve complex problems, nor can they do much at all on their own. In larger numbers however, they can find the shortest path to food and make sure that others are more likely to take that path, paving way for a system with which complex tasks are solvable[4].

The way ant colonies work together inspired Dorigo to write his thesis and is the basis behind the idea of artificial ants as a strong metaheuristic. Just as normal ants can solve complex problems together, artificial ants can be put towards creating good approximations for complex problems in manageable polynomial time. After a starting point is chosen, the ants traverse the problem space and take into consideration pheromone trails as well as other problem specific weights when picking their solutions. For instance, a weight within the Traveling Salesman Problem would naturally be the distances to the next available cities. When a solution is found, the ants would deposit pheromones and make that solution more likely to be explored by other ants in the future. Slowly, pheromones will build up on good, shorter paths while they slowly degrade along the less favorable paths. This will in turn lead to ants traversing the problem space closely around the best solutions and finding even better solutions as the algorithm creeps closer to stagnation[5].

procedure *ACO algorithm*
  Initialize pheromones
  while termination condition not met **do**
    ConstructAntSolutions
    ApplyLocalSearch % *optional*
    UpdatePheromones
  end
end

Figure 2.1: General pseudo-code for the ACO algorithm. Implementation for *ConstructAntSolutions* will depend heavily on the problem being solved, while *UpdatePheromones* will depend on the type of Ant System being used.

One problem that comes with this algorithm is stagnation. When many ants travel in the problem space they will put down a lot of pheromones, but as many of them gravitate towards the best solution they find, it

creates starvation in other parts of the dataset. This means that at some point, all ants will end up taking approximately the same path since the probability to veer off that path becomes negligible. This is due to the fact that pheromones degrade over time and only traveled paths get replenished, leading to situations where no better solutions can be found even if there is one. When this happens true stagnation is met, making it impossible to find further optimization. To combat this, a number of specialized algorithms have been engineered, all of them solving the stagnation problem in different ways. Following are the three ant systems we selected and implemented for this paper.

2.5.1 Elitist ant system

The biggest change that was made when EliteAS was created is the new set of ants present. These specialist ants functions differently compared to their normal counterparts. After all ants have constructed a solution, they normally deposit an amount of pheromones along their found paths. With EliteAS however, there are bonus pheromones given to the best found solution, essentially multiplying the amount of pheromones given by the number of specialist ants[2]. Every time a good solution is found it will get a massive influx of pheromones making it relevant for longer. Whilst normal paths degrade fast and become neglected, the elitist paths have high enough pheromone concentration to stay around as valid options. Therefore, just one deviating good path is enough to diverge from the problem space being currently searched and consequently preventing stagnation.

2.5.2 Rank-based ant system

RankedAS also use a form of specialist ants, but here they spread out pheromones on several good paths rather than giving it all to the best path found. Every path gets ranked according to its length, with the best ranked path getting the most pheromones and the worst path getting the least. Another aspect to RankedAS is that if there are fewer specialists than normal ants, the worst ranked paths will not get any pheromones at all[2]. For example with three specialists and ten normal ants, the ten ants would each find their own path while the spe-

cialists have the job to rank those paths and select the three best ones to give the pheromones to.

2.5.3 Min-max ant system

In the last of the systems that we implemented there are no specialist ants, going back to the old system of only using normal ants. The big change in MMAS on the other hand is how the pheromone levels are handled, introducing minimum and maximum values. The pheromone value on a path is not allowed to exceed the maximum or get lower than the minimum value. As there is a cap on both ends, a path's pheromones can never drop so low that the path becomes irrelevant. Similarly, a path will never get so saturated that it overshadows all other paths[10].

Added to this is a mechanic of smoothing. When one or more paths near the maximum value and all other paths near the minimum value, the pheromone levels are smoothed. This has an effect of greatly promoting paths with low pheromone levels, while still maintaining the standings between the paths[10]. Lastly, pheromones are only applied to the best path, ignoring all others. All of these additions to the basic ant system makes for a strong, competitive algorithm.

Chapter 3

Method

In this section we describe not necessarily our actual program code, but the formulas the algorithms are based on. We also mention how we decided to test the algorithms.

3.1 Problem space

When solving for TSP there are only two factors that affect the solution: the number of cities in the TSP instance and the distances between them. To make sure that the data acquired was varied and gave a good representation of the quality of answers, a few decisions were made. Firstly, all chosen TSP instances were problems with already known optimal solutions, so that a comparison between found solution and optimum could be made. Secondly, only symmetric TSP instances were used, meaning that the distance between city A and city B always is the same, regardless of which direction an ant is traveling. With asymmetric TSP there would need to be two layers of pheromones to prevent ants from getting tricked into traveling an edge that is only good from the opposite direction, needlessly complicating the problem. The three instances solved against were eil101 with 101 cities, tsp225 with 225 cities, and att532 with 532 cities.

3.2 Ant system implementations

All of the ant systems work in a similar way, following the same overall structure. The difference between them lay exclusively in the workings of pheromones, particularly how they are updated and maintained each iteration. In all of the ant systems, every ant that is present and working toward the solution will start out in a randomly selected city at the start of each iteration. When selecting the next city to travel to, the probabilities of each city are governed both by the amount of pheromones leading to it and its distance according to the following formula [2, 10] :

$$c_{ij}(t) = [\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta} \tag{3.1}$$

$$P_{ij}^{k}(t) = \frac{c_{ij}(t)}{\sum_{J \in N_i^k} c_{iJ}(t)}, \quad \forall j \in N_i^k \tag{3.2}$$

Eq. 3.1 describes how the heuristic value is calculated for edge (i, j). $\tau_{ij}(t)$ is the strength of the pheromone trail on edge (i, j) at iteration t, while η_{ij} is the heuristic data. In the case of TSP, $\eta_{ij} = 1/d_{ij}$ where d_{ij} is the distance between city i and j. Parameters α and β decide the relative influence that the pheromone trail and heuristic data should have. $\alpha = 0$ would mean that only the heuristic data is utilized, while $\beta = 0$ would mean the opposite[9].

Eq. 3.2 uses the heuristic value from Eq. 3.1 to calculate the probability that ant k would choose city j when currently located in city i. Here N_i^k holds all the cities that ant k yet has to visit[10].

The same articles [2, 10] also consider the reduction of pheromones each iteration. Ants in real life will only follow fresh trails laid out by their peers, while the pheromones on rarely used paths slowly disappear with time.

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + f(t) \tag{3.3}$$

The analog to this within the ACO algorithms is the constant ρ, a number between zero and one. Each iteration, all pheromone values are multiplied by this factor before the new additional levels of pheromones are added to the paths. This ensures that older paths not part of the best solution are slowly phased out from consideration. The function $f(t)$ in Eq. 3.3 refer to the specific pheromone update tech-

niques of each ant system, further explained in the following three sections.

3.2.1 Elitist ant system

EliteAS is the first of two ant systems we have chosen that relies on secondary, specialist ants to function. The presence of the specialist ants are needed to enforce the so-called elitist strategy. The pheromone update for EliteAS is defined as:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} + \sigma\Delta\tau_{ij}^{best} \tag{3.4}$$

$$\text{where} \quad \Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ travels on edge } (i,j) \text{ in its tour.} \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

$$\text{and} \quad \Delta\tau_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{if edge } (i,j) \text{ is part of the shortest tour found.} \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

where m is the number of normals ants, and σ the number of elitist ants (specialists). L_k is the length of the tour constructed by ant k and L_{best} is the length of the shortest tour found throughout all iterations. Q is instead a constant multiplier that simply dictates how much pheromones should be put down by all ants[2].

Eq. 3.5 describes how normal ants alter the pheromones while Eq. 3.6 describes elitist ants. Eq. 3.4 then adds together the result of all normal ants while multiplying the elitist ants.

3.2.2 Rank-based ant system

The largest difference between RankedAS and EliteAS is that normal ants do not directly add pheromones. RankedAS also drastically change how the specialist ants function. In contrast to the elitist ants, the ranked ants instead sorts all tours, giving each tour a rank, and then adds pheromones accordingly. The highest ranked tour being the shortest, and the rest increasing in length. Solutions outside the σ number of

best solutions found do not get additional pheromones, although they will still degrade [2]. Expressed as a formula we get the following:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{\mu=1}^{\sigma} \Delta\tau_{ij}^{\mu} \tag{3.7}$$

$$\text{where } \Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma - \mu + 1)\frac{Q}{L_\mu} & \text{if the ant with rank } \mu \text{ travels on edge } (i,j). \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Eq. 3.8 describes how the ranked ants affect the pheromones, with $\Delta\tau_{ij}^{\mu}$ being the change in pheromones by the ant with rank μ. The actual ranking of tours is not expressed. $(\sigma - \mu + 1)\frac{Q}{L_\mu}$ ensures that the ants with higher rank (closer to 1) give more pheromones than lower ranked ants. Since Eq. 3.7 sums the ranked ants up to σ, in the cases where $\sigma < m$, some tours will be ignored.

3.2.3 Min-max ant system

MMAS differs largely from EliteAS and RankedAS since it does not utilize specialist ants. Instead, its focus lies in adding a minimum and maximum value for pheromone concentration [10]. The updating of pheromones for MMAS is defined as follows:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij}^{best} \tag{3.9}$$

$$\text{where } \Delta\tau_{ij}^{best} = \begin{cases} \frac{Q}{L_{best}} & \text{if edge } (i,j) \text{ is part of the shortest tour found.} \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$

Eq. 3.10 is identical to the elitist strategy shown in Eq. 3.6, with the difference being that there are no specialist ants in Eq. 3.9 that amplify the result. Amplification is not needed since only the best tour gets updated. In EliteAS, L_{best} was defined as the length of the shortest tour found throughout all iterations. In our implementation of MMAS however, we use a combination of the global-best (L_{gb}) and iteration-best tour length (L_{ib}). More specifically, we use L_{gb} every 10th iteration. This is to provide the ants some guidance, preventing them from only searching around the iteration-best solution that may have veered far from the global-best solution.

Every time L_{gb} changes (whenever a new best solution is found), the minimum (τ_{min}) and maximum (τ_{max}) pheromone levels are updated according to:

$$\tau_{max} = \frac{1}{1-\rho} \frac{Q}{L_{gb}} \tag{3.11}$$

$$\tau_{min} = \frac{\tau_{max}(1 - \sqrt[n]{P_{best}})}{(avg - 1)\sqrt[n]{P_{best}}} \tag{3.12}$$

where n is the amount of cities and $avg = n/2$ is the average amount of cities an ant has to choose among during each step of tour construction. τ_{max} is an estimation of the maximum possible pheromone value, derived from the pheromone update formula (if L_{gb} also was the optimal tour length, it would no longer be an estimation)[10]. In Eq. 3.12, $\sqrt[n]{P_{best}}$ denotes the probability of choosing an edge part of the best solution after reaching stagnation. As follows, P_{best} is the probability of constructing the current best solution after stagnation. P_{best} is a constant parameter that has to be set, with lower values being preferred since that will increase the difference between τ_{max} and τ_{min}[10].

For MMAS we define stagnation as the state in which all edges of the best tour have reached τ_{max}, and all edges not part of that tour have fallen to τ_{min}. In this state, it is very unlikely for any ant to choose an edge not part of the best tour (although there is still a chance since $\tau_{min} > 0$), resulting in all ants constructing the same tour, thus stagnation. Following this, we can define the stagnation value γ as:

$$\gamma = \frac{\sum_{\tau_{ij} \in T} min(\tau_{max} - \tau_{ij}, \tau_{ij} - \tau_{min})}{n^2} \tag{3.13}$$

where T is the set of all pheromone trails and n is the amount of cities [6]. Essentially, the numerator will near zero as all trails near either τ_{min} or τ_{max}. Since γ takes some time to compute and does not vary much from iteration to iteration, we only check for stagnation every other iteration. This means that the operation of smoothing sometimes will be delayed by one iteration which is not significant.

When γ nears zero and MMAS thus has stagnated, all pheromone trails are smoothed according to:

$$\tau^*_{ij}(t) = \tau_{ij}(t) + \delta(\tau_{max}(t) - \tau_{ij}(t)) \tag{3.14}$$

where $\tau^*_{ij}(t)$ is the pheromone concentration after smoothing (still being part of the same iteration), and δ is the smoothing parameter [10]. If $\delta = 0$, the smoothing would have no effect, and if $\delta \geq 1$, every pheromone trail would be reset to τ_{max}. However, by setting δ to a value between 0 and 1, it is possible to greatly increase the pheromone concentration on less used paths, while still keeping the gathered heuristic information from previous iterations intact in relation to each other. That is, an edge that has less pheromones than another edge will still have less pheromones after smoothing and vice versa.

3.3 Test methodology

When testing, we wanted results that were granular enough to see the nature of the algorithms, while also being reasonably fast to compute. Thus the amount of ants were split into groups of 1, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100% of the amount of cities. The amount of specialist ants were instead split into groups of 1, 25%, 50%, 75% and 100% of the amount of regular ants. Five tests were performed for each combination of ants, specialists and TSP instance. These five tests were then combined to calculate an average result for each combination. Further, the same five predetermined seeds were used across all combinations. This simplified testing and ensured that the algorithms would experience the same randomness when the different input data did not have a big effect.

B. Bullnheimer [2] mentions in their paper that a constant of 5 ranked ants gave good results with their tests of RankedAS. Because of this, an extra sixth group of specialists were tested specifically for RankedAS, where $\sigma = 5$.

In Fig. 2.1 from Section 2.5 there is a while-loop with a condition that dictates how long the ants will search for the optimal path. In our tests, we decided to loop for $20000/m$ iterations, where m is the amount of ants used. Since m paths are constructed each iteration, this means that approximately 20000 paths are constructed in total per test which should give a fair comparison.

Since the goal of this thesis was to find how the number of ants changed the length of solutions, all other parameters were kept constant be-

tween all tests. Further, the values were set to the same values proposed in other papers, as to not spend time trying to optimize them. The general parameters used in all ant systems were set to:

$$\alpha = 1, \quad \beta = 5, \quad \rho = 0.5, \quad Q = 100$$

as proposed in [2,3,5,7]. The parameters unique to MMAS were instead set to:

$$\delta = 0.99, \quad P_{best} = 0.005$$

as proposed in [10]. Also, when checking for stagnation, the pheromones were smoothed whenever the stagnation value γ dipped below 10^{-12}.

Chapter 4

Results

In this section the results after running all tests on EliteAS, RankedAS and MMAS are presented. All figures show how the deviation from optimum changes with the number of ants used. A lower value on the y-axis is preferred.

4.1 Elitist ant system

Something that is visible in all three graphs, but most clearly seen in Figure 4.1, is that the number of specialists has a large impact on the solution. The fewer specialists there are, the worse the solution is compared to the known optimum. However, the number of normal ants do not affect the solution as much. The best solution is found in all three test cases with 100% elitist ants and around 10% normal ants.

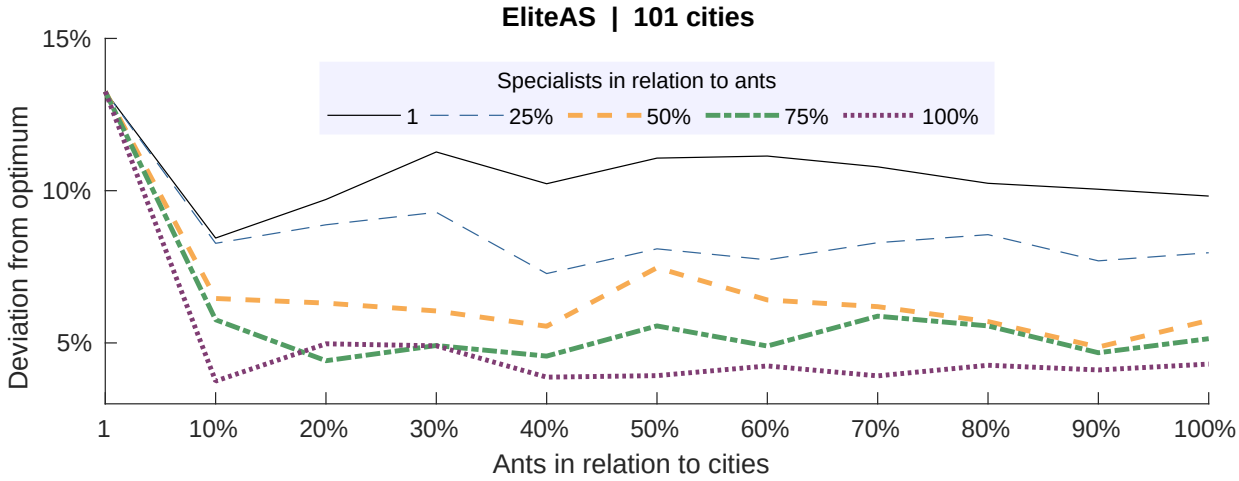

Figure 4.1: Average deviation from optimum using EliteAS solving eil101.

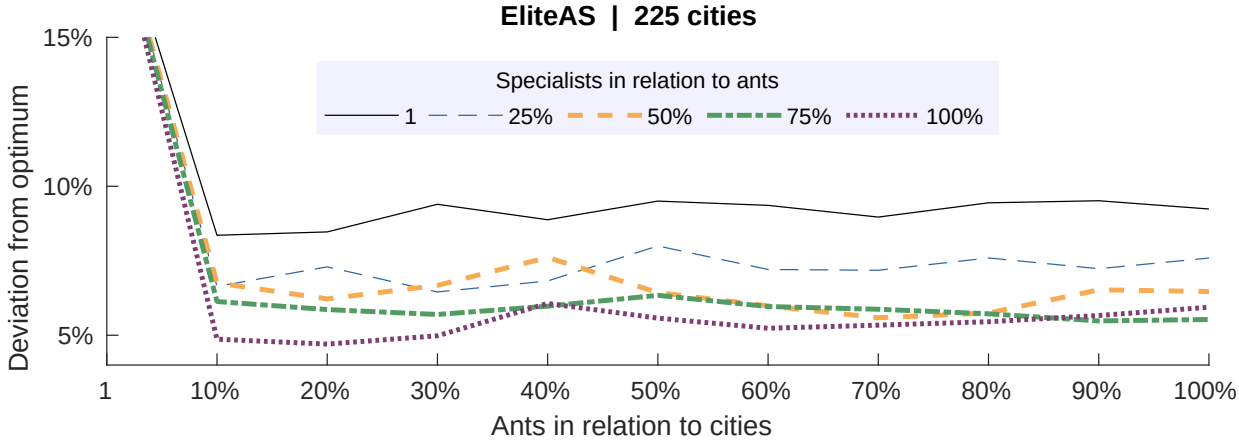

Figure 4.2: Average deviation from optimum using EliteAS solving tsp225.

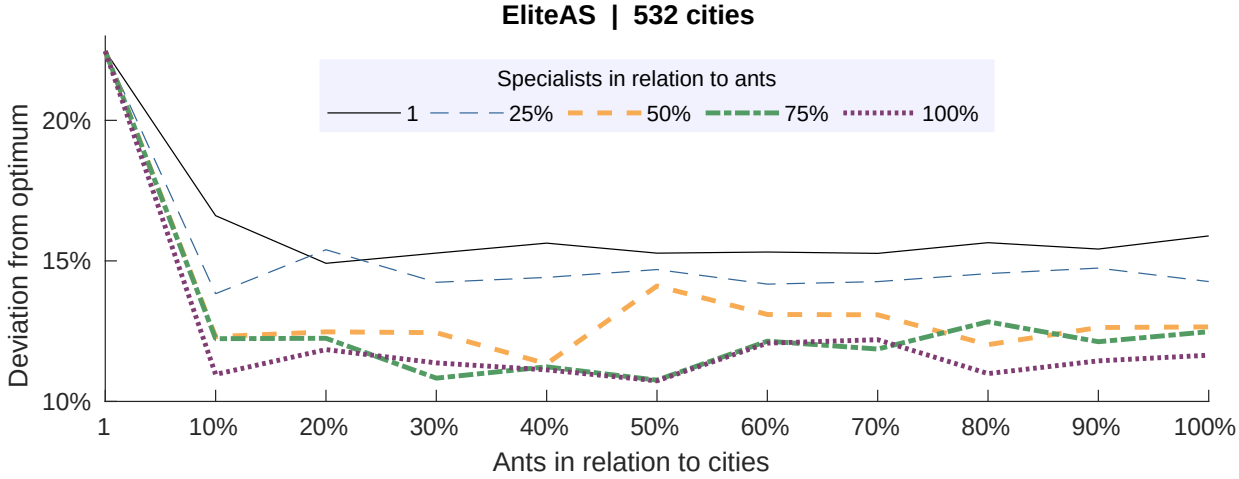

Figure 4.3: Average deviation from optimum using EliteAS solving att532.

4.2 Rank-based ant system

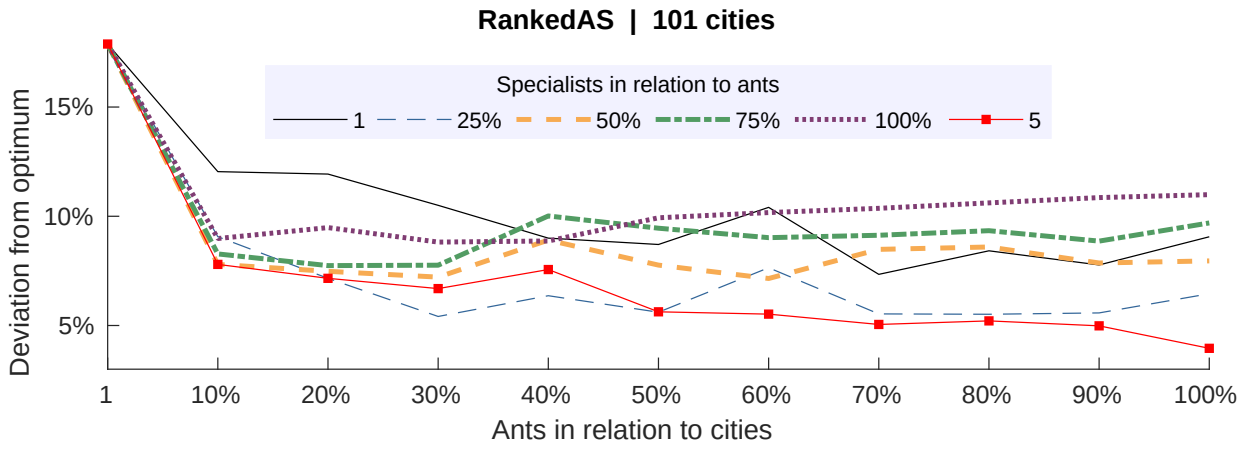

Figure 4.4: Average deviation from optimum using RankedAS solving eil101.

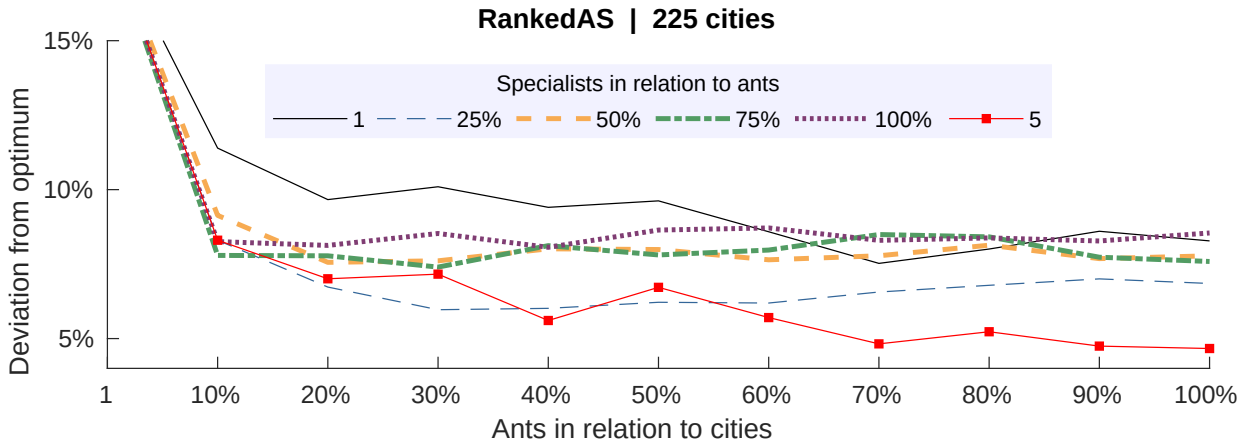

Figure 4.5: Average deviation from optimum using RankedAS solving tsp225.

The results of RankedAS are opposite that of EliteAS. A large percentage of specialist ants makes the solution worse. The amount of normal ants has a noticeable effect when using 5 ranked ants, but not when using more. The best solution is found with 5 specialists and 100% normal ants in relation to cities. However, Figure 4.6 displays a deviation where using more than 50% normal ants worsens the solution.

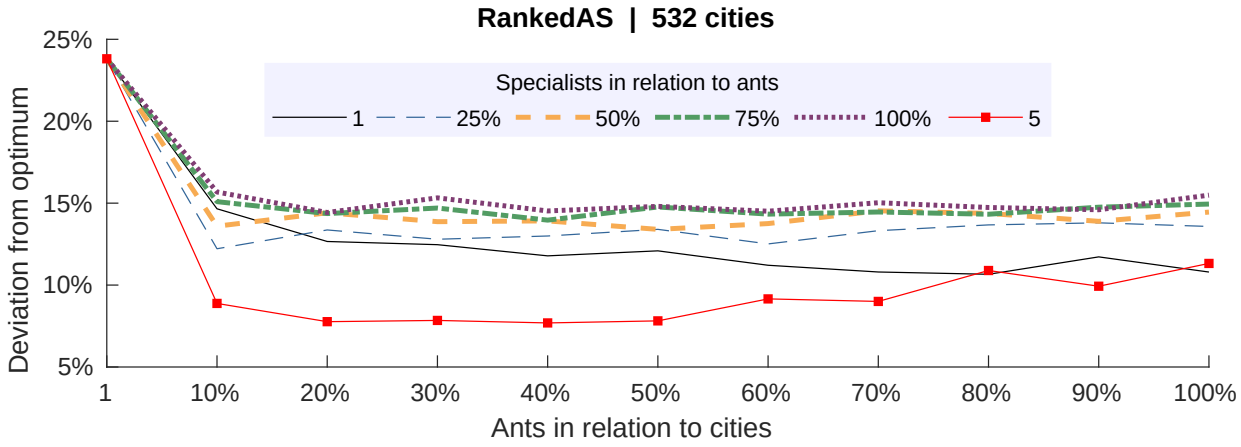

Figure 4.6: Average deviation from optimum using RankedAS solving att532.

4.3 Min-max ant system

The final ant system, MMAS, does not utilize specialist ants. Instead of showing the different specialist ant categories, the blue area in the graphs showcase the variance of the solutions (as mentioned in Section 3.3, five tests are performed for each test case). The solutions in

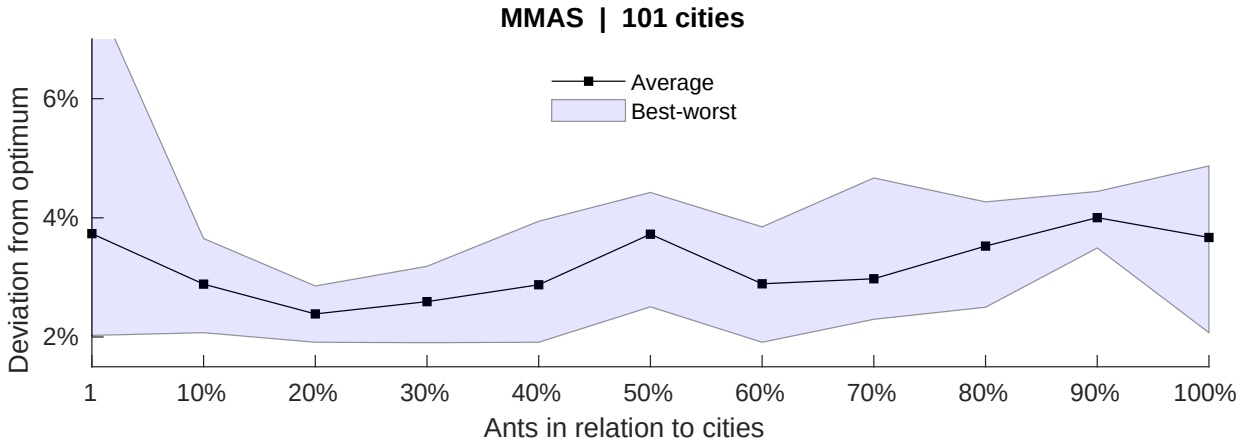

Figure 4.7: Average deviation from optimum using MMAS solving eil101.

Figure 4.7 and Figure 4.8 vary between 2 and 4 percent while they vary between 6 and 11 percent in Figure 4.9. The best solutions are found within the span of 10-30% ants in relation to cities across all three test cases.

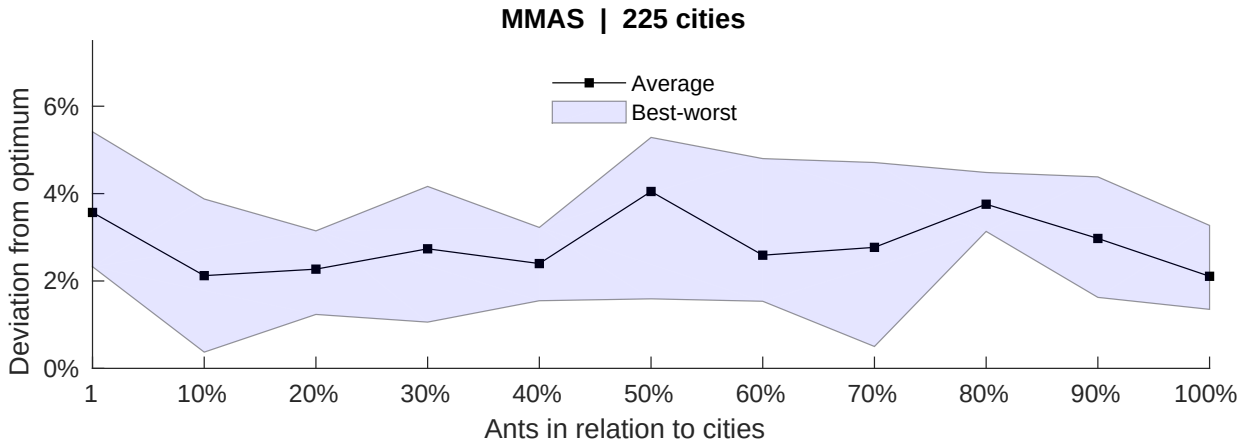

Figure 4.8: Average deviation from optimum using MMAS solving tsp225.

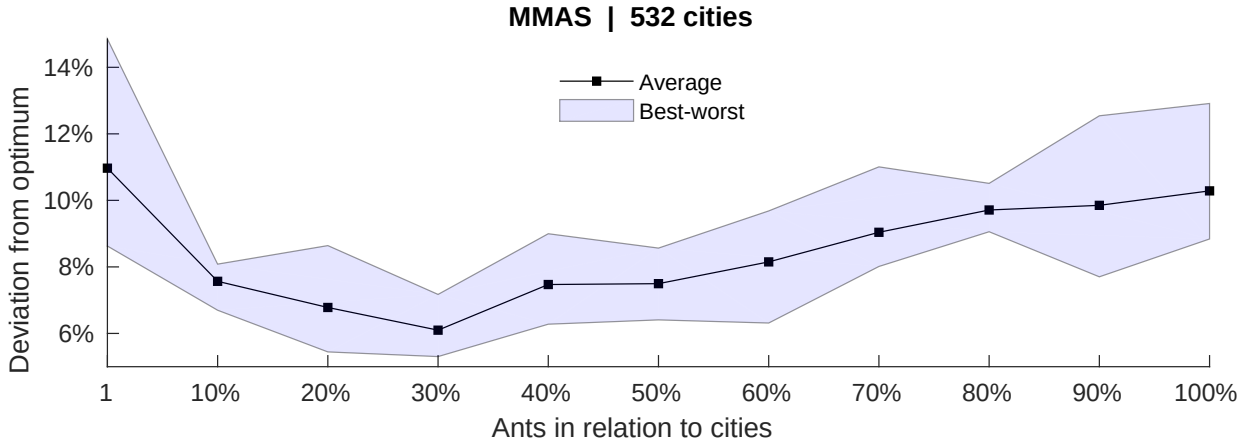

Figure 4.9: Average deviation from optimum using MMAS solving att532.

Chapter 5

Discussion

In this section we discuss the results and how the different ant systems compare to one another. We also discuss how we could improve on the data found and any flaws in the methods used.

5.1 Result discussion

Something that came to light with the results was that the number of normal ants used had less impact than anticipated. Although, this result is not entirely unexpected. How pheromones are implemented and updated between iterations is what mainly differentiate the ant systems. Therefore, it is not strange that the highest impact on tour length is tied to pheromones and the values that manipulate them, namely specialists.

When EliteAS updates the pheromones there are many small updates, one for every constructed tour, with the best solution getting multiplied by every elitist ant. When this multiplier is small or close to 1, the best solution gets next to no emphasis. In that case, there is no motivation for the ants in future iterations to search around the best found tour instead of over any of the other tours. This is mirrored in the data where more specialists, giving a higher multiplier, result in the best solutions. With high numbers of specialists the amount of added pheromones is so large that it take considerable time to dissipate the concentrated pheromones, consequently making the path at-

tractive considerably longer. Non-optimal tours dissipate much faster than the best tours of previous iterations, making it so only few tours truly gain traction. Therefore, there is no need for a massive amount of normal ants to find solutions as the individual impact they have is negligible compared to the specialists. The data found in our tests suggest that 10% ants in relation to cities, and as many specialists as ants generally give good results for EliteAS.

A similar trend was observed with MMAS as well. The number of ants present within the solution did not change the end result in any meaningful way except with Figure 4.9 which had the highest fluctuations in solution length. Even with the fluctuation present, MMAS always performed better than its counterparts in all test cases. The introduction of lower and upper limit on pheromones lowers the risk of stagnation, ensuring that no edges ever truly die out. Even when stagnation occurs, pheromone smoothing revitalizes all edges while still preserving the data gathered in previous iterations. After smoothing, there is a higher probability that edges close to best solution that were not considered before, now becomes targeted by the ants. Even so, the main reason why many ants do not perform better is because only one tour affects the pheromones each iteration. With our implementation, the number of ants directly influence the amount of iterations; more ants mean fewer iterations. Thus, there is no real benefit to having a large amount of ants as this gives MMAS less time to converge towards the best solution. Although the data only show minor differences, the span of 10-30% ants in relations to cities consistently give good results. That span gives a good balance between number of iterations and number of ants.

With RankedAS there were signs of the same trends as the other ant systems. In many cases the number of ants had little impact on the overall performance. However, when it came to specialists the result was opposite that of EliteAS. In RankedAS the result got markedly worse as more specialists were used, mirroring the behavior seen in EliteAS. The biggest differentiator was the special test case with five ranked ants. The solutions were better when many ants were used in conjunction with five ranked ants, showing best results at 100% ants. This is most likely tied to the fact that the amount of ranked ants directly affects how many tours are included in the pheromone update. With many ranked ants, edges in solutions might overlap. Even if sub-

optimal tours are updated with less pheromones than optimal tours, there is a risk that the overlap creates a concentration of pheromones. This concentration factors into the probabilistic choice of cities, diluting the quality of the cities with high probability. Inversely, when only few ranked ants are used, there is value in having many normal ants working towards the solution. With more normal ants, the likelihood to find good tours increases. Subsequently, more ranked ants will get good tours. This results in a good and wide search space the next iteration, increasing the chance of finding near-optimal solutions.

That being said, with the last test case, Figure 4.6 showed a different result. In this case the trend of getting better results for larger amount of ants with five specialists was not realized, instead showing the reverse. Data like this was surprising, but in retrospect is is not that strange. As mentioned, the number of iterations is capped and is lower when more ants are present. The ants simply do not have enough time to properly compute the answer, the solution quality suffering as a result. In Figure 4.5 with 225 cities, 100% ants gave the best solution, only being slightly more optimal than 90%. In Figure 4.6 with 532 cities, 40% and 50% ants gave roughly the same answer, with more ants giving progressively worse results. This shows that approximately 225 ants is the turning point for our implementation of RankedAS.

5.2 Possible improvements

One problem that was faced when compiling data for this thesis was the fact that time and resources were limited. To make sure that deadlines were met, as well as making sure that weeks where not spent testing excessively, concessions had to be made. Capping the number of iterations that the ants had to construct solutions was the most logical and practical answer to stop runaway execution times. With the limitations in place, it still took more than a day to compute all the data, using multiple computers running multiple instances of the test program.

Limiting the iterations had some side effects that did show in the final data, most notably in the case of RankedAS with five ranked ants and a large number of ants. For future testing on this subject, we firmly believe that larger data sets have the same form of performance as

their 101 and 225 city counterparts if given enough time to compute. Another factor of note is the the values used for parameters in the algorithms were taken from other papers at face value. The performance of the implementations made for this thesis could theoretically improve if there was more time to experiment and see how parameters aside from ants change computed solutions. Future studies that can give the ACO algorithms time to properly converge without the time constraints present for this thesis will get better results, as well as opportunity to find more optimal values for other input data.

Chapter 6

Conclusion

After implementing three variations of the ACO algorithm, namely EliteAS, RankedAS and MMAS, we studied the effect that the number of ants has on computed results. Generally, the amount of normal ants had little impact on the overall performance, although the impact was mostly consistent between test instances. For example, with RankedAS using five specialists there was a large increase in performance with 100% ants in relations to cities used. This is in contrast to EliteAS and MMAS where lower amount of ants, around 10-30%, gave better performance. The conclusion derived from this suggests that the optimal amount of ants is heavily reliant on the implementation, and there is not one universal answer. Regarding specialist ants, EliteAS consistently produced favorable results with many elitist ants while RankedAS instead had better results with lower amounts of ranked ants, five ants being the best. This again showing that the optimal number of ants vary with each algorithm. More testing of other implementations of the ACO algorithm is needed and this thesis should be a good stepping stone for further research.

Bibliography

[1] *2nd Workshop on Biological Distributed Algorithms*. http://www.cs.cmu.edu/~saketn/BDA2014/. Austin, Texas, 2014.

[2] C. Strauß B. Bullnheimer R. Hartl. "A new rank based version of the Ant System. A computational study". In: *Working Papers SFB "Adaptive Information Systems and Modelling in Economics and Management Science"* (1997). http://epub.wu.ac.at/616/.

[3] M. Dorigo. "Optimization, learning and natural algorithms". In: *PhD Thesis, Politecnico di Milano* (1992).

[4] LM. Gambardella M. Dorigo. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". In: *IEEE Transactions on Evolutionary Computation* (1997), pp. 53–66.

[5] LM. Gambardella M. Dorigo. "Ant-Q: A Reinforcement Learning approach to the traveling salesman problem". In: *Proceedings of the Twelfth International Conference on Machine Learning* (1995). https://www.sciencedirect.com/science/article/pii/B9781558603776500396, pp. 252–260.

[6] T. Stützle M. Dorigo. *Ant Colony Optimization*. A Bradford Book, 2004.

[7] T. Stützle M. Dorigo. *Ant Colony Optimization: Overview and Recent Advances*. Tech. rep. http://iridia.ulb.ac.be/IridiaTrSeries/rev/IridiaTr2009-013r001.pdf. Université Libre de Bruxelles, 2009, pp. 351–357.

[8] A. Lewis M. Randall. "A Parallel Implementation of Ant Colony Optimization". In: *Journal of Parallel and Distributed Computing* (2002).

[9] N. Slamy S. Yaseen. "Ant Colony Optimization". In: *International Journal of Computer Science and Networks Security* (2008). http://paper.ijcsns.org/07_book/200806/20080649.pdf, pp. 351–357.

[10] H. Hoos T. Stützle. “MAX-MIN Ant System”. In: *Future Generation Computer Systems* (2000). https://www.sciencedirect.com/science/article/pii/S0167739X00000431, pp. 889–914.