An Analysis of the Ant Colony Optimization Parameter Space

Javier Bosch

4th Year Project Report
Computer Science and Mathematics
School of Informatics
University of Edinburgh

2023

Abstract

This dissertation investigates the performance of the Ant Colony Optimization (ACO) algorithm in solving the Travelling Salesperson Problem (TSP), with a focus on the exploration of the parameter space. A simple, open-source Java implementation of the ACO algorithm is developed and used throughout the project. The impact of key parameters, such as the evaporation rate (ρ) and others, on the algorithm's efficiency and effectiveness, is examined in-depth.

The results reveal a functional dependence between the evaporation rate and the number of iterations, which can be used to optimize the ACO algorithm's performance. The potential of a dynamic evaporation rate is also explored, showing promising results. The applicability of the ACO algorithm to real-world optimization problems is discussed, with urban waste management as an illustrative example. This research contributes valuable insights for the development and improvement of ACO in solving complex optimization problems.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics Committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Javier Bosch*)

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Michael Herrmann, for his invaluable guidance and support throughout my dissertation. Without his expertise and dedication, this work would not have been possible. Thank you to Angus Murdoch and Gareth Barwell from The City of Edinburgh Council for providing me with some information about Edinburgh. I also want to give a special shoutout to Helene, whose sharp eye for detail and tireless efforts in proofreading greatly improved the quality of this text.

Finalmente, quiero expresar mi agradecimiento a mi familia y amigos, en especial a mi padre, mi madre y mi hermana, por su apoyo inquebrantable durante estos últimos cuatro años. Su amor y ánimo han sido mi fuente de inspiración.

Table of Contents

Chapter 1

Introduction

Metaheuristic algorithms are a valuable tool for solving complex and computationally demanding optimization problems. Unlike exact algorithms, metaheuristics do not guarantee the discovery of an optimal solution but provide near-optimal or *good-enough* solutions within a reasonable amount of time. The value of metaheuristics lies in their ability to tackle very challenging or NP-hard problems, such as combinatorial optimization problems. By employing intelligent search strategies that involve exploration and exploitation, metaheuristics can efficiently navigate large search spaces and find high-quality solutions [41].

The Travelling Salesperson Problem (TSP) is a classical NP-hard optimization problem that has been extensively studied in the field of computer science. Given a set of cities and the distances between each pair, the goal is to find the shortest possible route which visits each city once before returning to the starting city. TSP has numerous real-world applications, such as vehicle routing, logistics, and circuit board manufacturing, among others. Due to its computational complexity and practical relevance, TSP has become a benchmark problem for evaluating the performance of various optimization algorithms, including metaheuristics like Ant Colony Optimization (ACO), Genetic Algorithms (GA) and Particle Swarm Optimization (PSO).

In the context of TSP, metaheuristics offer a viable means of finding near-optimal solutions in a computationally efficient manner. ACO, in particular, has demonstrated promise in tackling TSP and other optimization problems.

1.1 Motivation

ACO is an elegant and intuitive metaheuristic optimization algorithm that garnered significant attention during the 1990s and early 2000s. Introduced by Marco Dorigo in his seminal PhD thesis (1992) and further developed in collaboration with numerous researchers in the field [15]. ACO is inspired by the natural behaviour of ants searching for food. The algorithm mimics the ants' ability to find the shortest path between their nest and a food source by using pheromone trails as a form of communication [21]. The ACO algorithm has since undergone several transformations, with various

modifications and improvements proposed to enhance its performance and applicability. The versatility of ACO has led to its successful application in numerous real-world domains, such as telecommunications [54], logistics [13], and manufacturing [45], among others[26].

Despite its simplicity and ease of implementation, research interest in ACO has dwindled in recent years. Most of the existing literature on ACO focuses on introducing new parameters, developing hybrid algorithms, and applying ACO to various problem domains[26]. These include the Vehicle Routing Problem (VRP), Quadratic Assignment Problem (QAP), and others ([22, 25]). Despite the shift of academic interest from ACO to other optimization techniques, such as Reinforcement Learning [59], ACO still holds promise for addressing complex combinatorial optimization problems.

One of the challenges in applying ACO to real-world problems is finding and selecting critical values for its parameter, such as pheromone evaporation rate and heuristic bias. These parameters significantly affect the algorithm's performance as they determine the balance between exploration and exploitation [10]. Identifying the optimal parameter settings remains an open question in ACO, which has motivated researchers to refine and improve the algorithm over the years continuously.

In contrast to other metaheuristic algorithms, such as Particle Swarm Optimization (PSO) [49, 28], which are thoroughly investigated and understood, there is still a lack of comprehensive understanding of ACO's parameters and their critical values in the scientific community. This dissertation aims to address this knowledge gap by conducting an in-depth study of ACO parameters and their implications on the algorithm's performance.

1.2 Project Goals and Contributions

The primary goal of this project is to achieve a deeper understanding of the parameters involved in ACO and to identify their critical values, while simultaneously developing an open-source implementation of the ACO algorithm. The aim is to provide insights that can improve the performance of ACO algorithms by examining the behaviour and significance of its parameters. Additionally, this dissertation will demonstrate a real-world application of ACO by solving a routing problem in The City of Edinburgh, showcasing the practical utility of the insights gained.

This dissertation aspires to revitalize interest in ACO as a powerful optimization technique by providing a comprehensive and systematic analysis of its critical aspects, alongside the development of an open-source implementation, laying the groundwork for future research and innovation. By examining the algorithm in this context, the hope is that this work will inspire new ideas and further applications of ACO in real-world scenarios.

1.3 Project Structure

This dissertation is organized into several chapters, each focusing on different aspects of the project:

- **Introduction**, provides an introduction and motivation for the project, outlining its goals and contributions.
- **Background**, presents the necessary background information, including the biological inspiration for ACO, the development of various ACO variants, and the theoretical bounds for the TSP.
- **Methods**, describes the methodology employed in this study, detailing the experimental setup for the process of investigating the critical values of ACO parameters.
- **Results**, presents the results and analysis of the experiments and a real-world application of ACO, where the algorithm is applied to a routing problem in The City of Edinburgh.
- **Discussion**, discusses the results of the experiments and highlights the key insights and relationships between ACO parameters and their impact on algorithm performance.
- **Conclusion and Further Study**, concludes the dissertation, summarizing the main findings, contributions, and potential directions for future research of ACO.

Chapter 2

Background

2.1 Biological Inspiration

In the mid-20th century, French entomologist Pierre-Paul Grassé made a series of observations on the behaviour of select termite species. He noticed that these insects responded to specific environmental stimuli, which influenced their actions and those of other individuals in the colony. Grassé coined the term ”stigmergy” to describe this unique form of communication, characterized by the fact that insects are stimulated by the outcomes of their actions [34].

2.1.1 Stigmergy

Two main features distinguish stigmergy from other forms of communication:

- Stigmergy is an indirect, non-symbolic communication mediated by the environment. Insects exchange information by modifying their surroundings.
- Stigmergic information is localized, meaning that it can only be accessed by insects that visit the specific location where the information was released or its immediate vicinity.

The concept of stigmergy is critical to understanding the collective intelligence of social insects, such as ants, bees, and termites. These insects exhibit complex behaviours that emerge from individual interactions, despite their relatively simple individual cognitive abilities. The study of stigmergy helps researchers gain insights into the mechanisms behind self-organization and swarm intelligence in social insects, which can be applied to artificial systems.

Ant colonies provide a compelling example of stigmergy in action. Many ant species deposit pheromones on the ground while travelling to and from a food source. Other ants sense the presence of these pheromones and are more likely to follow paths with higher pheromone concentrations, ultimately leading to highly efficient food transportation to the nest [34].

2.1.2 Double Bridge Experiment

Deneubourg and his colleagues conducted extensive research on the pheromone-laying and the behaviour of ants [17]. In a well-known ”double bridge experiment” (Figure 2.1) a colony of Argentine ants was connected to a food source via two bridges of equal lengths. In this setup, ants began exploring the area around their nest and eventually discovered the food source. As they travelled between the food source and the nest, the ants deposited pheromones along their paths. At first, each ant randomly chose one of the two bridges. Over time, random fluctuations lead to one bridge accumulating more pheromone trails than the other bridge, attracting more ants. This positive feedback mechanism led to even more pheromone deposition on the favoured bridge, ultimately converging the entire colony using the same bridge.

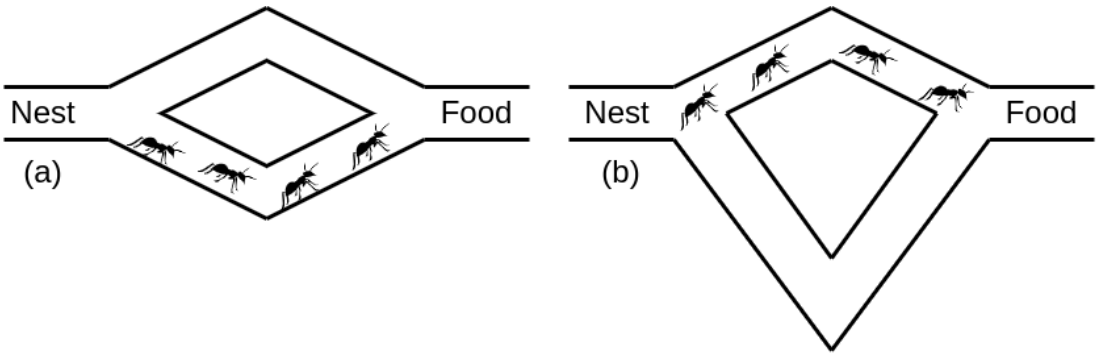

Figure 2.1: Experiment setup for ”double bridge experiment”. For (a), branches have the same length. For (b), branches have different lengths.

This experiment highlighted the potential of stigmergy as an efficient communication system and a powerful optimization process. By exploiting the collective intelligence and decentralized decision-making of the ant colony, the ants were able to find optimal or near-optimal solutions to complex problems in a relatively short period of time. This observation sparked interest in the scientific community, particularly in the fields of artificial intelligence and optimization, where researchers began exploring the possibility of translating these natural mechanisms into computational algorithms.

This colony-level behaviour, which relies on autocatalysis or positive feedback, enables ants to efficiently locate the shortest path between their nest and a food source. Goss et al. further investigated this phenomenon in a variant of the ”double bridge experiment”, where one bridge was significantly longer than the other [33]. In this case, random fluctuations in the initial bridge choice were less pronounced, and ants that initially chose the shorter bridge reached the nest more quickly. Consequently, the shorter bridge received pheromone deposits earlier, increasing the likelihood that additional ants would choose it over the long bridge.

2.1.3 Ant Systems

Drawing inspiration from these observations of ant behaviour and the concept of stigmergy, researchers sought to harness these biological principles to develop a novel

optimization algorithm. The ACO algorithm emerged as a population-based metaheuristic that simulates the behaviour of a colony of artificial ants to find optimal solutions to combinatorial optimization problems. ACO captures the essence of ant behaviour, such as pheromone deposition, sensing, and evaporation, and translates these natural phenomena into mathematical constructs [21].

In ACO, each artificial ant represents a potential solution to the problem at hand, and their collective behaviour imitates the stigmergic communication found in real ants. Artificial ants iteratively construct solutions by moving through a problem space, guided by a combination of heuristic information and pheromone trails. As they construct solutions, the ants deposit pheromone trails that correspond to the quality of their solutions, effectively communicating this information to other ants in the colony. Over time, this process leads the colony to converge towards high-quality solutions, with the best-performing ants influencing the search direction of the entire colony.

2.2 Ant Colony Optimisation

In this section, an overview of the development of ACO will be presented, its variants and recent advancements discussed and some notable applications explored.

2.2.1 First Example

The original algorithm [15], known as Ant System (AS), was designed to solve the TSP. The success of AS in tackling TSP sparked interest in applying ACO algorithms to other combinatorial optimization problems, such as the Quadratic Assignment Problem [56], the Vehicle Routing Problem [9], and the Scheduling Problem [37].

2.2.2 Standard Variants

Following the introduction of the original Ant System, several variants have been proposed to improve its performance and adapt it to different problem domains. Some of the most well-known ACO variants include:

Ant Colony System (ACS) [24]: Proposed by Dorigo and Gambardella in 1997, ACS introduces several modifications to the original Ant System, such as the use of a local pheromone update rule and the exploitation of an additional heuristic. The ACS was meant to enhance the exploration-exploitation balance and accelerate the convergence towards optimal solutions compared to the original algorithm.

Max-Min Ant System (MMAS) [58]: Developed by Stützle and Hoos in 2000, MMAS introduces an explicit pheromone trail limit, which helps to avoid premature convergence and improve the search performance. This variant also employs a more aggressive pheromone update strategy to promote exploring new solutions.

Rank-based Ant System (RAS) [12]: Introduced by Bullnheimer et al. in 1999, RAS employs a rank-based pheromone update mechanism that takes into account the quality of solutions found by individual ants. By rewarding high-quality solutions with stronger

pheromone updates, RAS encourages the search towards promising regions in the solution space.

2.2.3 Recent Advancements

Recent advancements in ACO research have focused on enhancing the algorithm's performance, adaptability, and applicability. Some of these advancements include:

Hybrid ACO [11, 48, 66]: Combining ACO with other optimization techniques, such as local search or other metaheuristics, has emerged as a popular approach to improve the algorithm's performance. These hybrid algorithms can exploit the complementary strengths of the combined techniques, often resulting in more efficient and effective solutions.

Dynamic and Multi-objective ACO [36, 3, 29]: Researchers have extended the ACO framework to handle dynamic and multi-objective optimization problems, where the problem's constraints, objectives, or environment may change over time or involve multiple conflicting objectives. These extensions often involve adapting the pheromone update mechanisms and incorporating additional heuristics to effectively tackle the increased complexity.

Parallel ACO [60, 64, 51]: To further improve the computational efficiency of ACO algorithms, researchers have explored parallel implementations that leverage the inherent parallelism found in the foraging behaviour of ants. These parallel ACO algorithms can be executed on multi-core processors, distributed memory systems, or even specialized hardware, such as GPUs, to significantly reduce the computation time required to solve large-scale optimization problems.

2.2.4 Applications

ACO algorithms have been successfully applied to a wide range of combinatorial optimization problems in various domains, demonstrating their versatility and effectiveness. Some notable applications include:

Transportation and logistics [31]: ACO has been employed in solving Vehicle Routing Problems, Fleet Management Problems, and other transportation-related optimization tasks, leading to more efficient routing and resource allocation. These applications contribute to reducing transportation costs, minimizing environmental impact, and improving overall operational efficiency.

Telecommunications [19]: In the field of telecommunications, ACO has been applied to problems such as network routing, channel assignment, and network design. By optimizing the use of network resources and minimizing communication delays, ACO algorithms can help enhance the quality of service and network reliability.

Scheduling [18]: ACO has been used to solve various scheduling problems, including job-shop scheduling, flow-shop scheduling, and timetabling problems. These applications contribute to improved resource utilization, reduced production times, and increased overall efficiency in industries and educational institutions.

Bioinformatics [43, 46]: In the realm of bioinformatics, ACO has been applied to problems such as protein folding, gene expression analysis, and multiple sequence alignment. These applications can help advance our understanding of complex biological systems and contribute to developing new drugs and therapies.

Overall, the Ant Colony Optimization algorithm has come a long way since its inception, with numerous variants and advancements contributing to its success in tackling a wide range of combinatorial optimization problems [26]. However, as with any optimization algorithm, it is crucial to critically evaluate its performance and applicability, considering the specific characteristics and requirements of the problem at hand. Researchers should continue to explore new ways to improve the algorithm, incorporate insights from other fields, and maintain scientific rigour in the development and evaluation of ACO.

2.2.5 Convergence Proof

The Convergence Proof for ACO establishes that ACO converges to the global optimum under certain conditions. In this proof from 2002 Stützle and Dorigo [57] show that ACO is guaranteed to find an optimal solution with a probability that can be made arbitrarily close to one if given enough iterations. As the iterations increase, the proof indicates that the likelihood of ants selecting the optimal path also increases, which leads to stronger pheromone levels on that path. Hence, there is a positive feedback loop, where ants are now more likely to choose the optimal path as the pheromone levels get stronger, and the pheromone levels continue to increase as more ants follow that path. This positive feedback mechanism eventually guides the algorithm's convergence to the global optimum.

This proof provides a greater theoretical foundation for ACO, as well as a better understanding of the algorithm. Similar metaheuristic algorithms lack this kind of proof. As a result, ACO is often preferred over other metaheuristic algorithms in applications where convergence to the global optimum is critical.

2.2.6 Criticality

Criticality is a concept that originates from the field of physics, particularly in the study of phase transitions and complex systems. In these contexts, criticality refers to the point at which a system undergoes a transition from one state to another, often characterized by significant changes in its properties and behaviour [40, 7]. The idea of criticality is often borrowed and applied to various other disciplines, including computer science, biology, and social sciences, to describe phenomena that exhibit similar transitional behaviours.

In the context of metaheuristics, criticality plays a crucial role in determining the algorithm's performance. It is closely related to the balance between exploration and exploitation during search, which is essential for avoiding premature convergence to suboptimal solutions and ensuring the discovery of high-quality solutions [27].

For instance, criticality has been studied and better understood in the context of Particle

Swarm Optimization (PSO) [44, 28]. Researchers have identified critical parameter values that govern the balance between exploration and exploitation in PSO, leading to improvements in the algorithm's performance and adaptability. These findings have provided valuable insights into the role of criticality in the search process and its implications on the algorithm's efficiency.

However, in the case of ACO, criticality has not been as thoroughly investigated, especially for real-world optimization problems. Most of the existing research on criticality in ACO has focused on toy examples, as demonstrated by Herrmann et al. [39]. This limited understanding of criticality in ACO has left a significant knowledge gap that needs to be addressed to optimize the algorithm's performance and adaptability.

2.2.7 Use of Local Search in ACO

The integration of local search techniques into ACO has been a significant area of research, as it can lead to improved solution quality and enhanced algorithm performance, for little computational cost. Many researchers, such as Stützle [58] and others [35, 61], have recognized the value of incorporating local search into ACO algorithms to refine and optimize the solutions generated by ants.

Local search methods, such as 2-opt or 3-opt moves, enable the exploration of the immediate neighbourhood of a solution and can effectively fine-tune the solutions generated by the artificial ants. By performing a series of small, localized modifications, the algorithm can converge more quickly to high-quality solutions while still maintaining the benefits of the global search process provided by the ACO framework [65].

To maintain a clear focus on the core ACO algorithm in this dissertation, we will primarily focus on ACO algorithms without the incorporation of local search techniques. While acknowledging the potential benefits of integrating local search methods, our primary objective is to provide a comprehensive understanding of the fundamental principles and mechanisms that drive the success of ACO algorithms.

2.3 Travelling Salesperson Problem

The TSP is a classical combinatorial optimization problem that aims to find the shortest possible route for a salesperson who must visit a set of cities and return to the origin city, ensuring that each city is visited exactly once. TSP is an NP-hard problem, which means that finding an exact solution for large instances of the problem is computationally intractable within a reasonable time frame. The difficulty of solving the TSP arises from the fact that the number of possible solutions increases factorially with the number of cities, leading to a combinatorial explosion as the problem size grows [4].

2.3.1 Theoretical Bounds

Given that the TSP is an NP-hard problem, finding optimal solutions for most of its instances is infeasible. This raises the question of how to determine whether a solution is *good enough* while analyzing the results of experiments in this dissertation. Fortunately,

for instances of the TSP that are confined within the unit square, it is possible to take advantage of established lower and upper bounds to assess the quality of an optimal TSP solution.

The theoretical **lower bound** for the TSP within the unit square was established by Beardwood, Halton, and Hammersley in 1959 [8]. They derived a formula that provides an asymptotic lower bound for the length of the shortest tour as the number of cities increases. This bound can be used to compare the solutions generated by optimization algorithms and to gauge their performance.

The **upper bound** for the TSP can be derived from various heuristic algorithms. For example, Fiechter's parallel tabu search algorithm, which is known to produce high-quality solutions, can be used to establish an upper bound for the problem [30]. Additionally, several approximation algorithms, such as the Christofides algorithm, can also provide an upper bound for the TSP with guaranteed performance ratios [38, 14].

By taking advantage of these theoretical bounds, we can evaluate the performance of ACO on instances of the TSP within the unit square. For the context of this dissertation, we can assume that,

$$\text{Lower bound} = 0.7078\sqrt{n} + 0.551$$

where n is the number of cities in the TSP instance, as shown by Christine L. Valenzuela and Antonia J. Jones in [62].

2.3.2 Ant Colony Optimization for the TSP

ACO has been widely recognized as an effective metaheuristic algorithm for solving various NP-complete optimization problems, including the TSP [24].

To apply ACO to the TSP, we first initialize a population of artificial ants. These ants are randomly placed in different cities and tasked with constructing complete tours of the given cities. The ants' decisions on which city to visit next are influenced by the distance between their current city and all other reachable cities and the pheromone trails previously laid by other ants.

After each iteration, each ant lays pheromones on the path it took inversely proportional to the total tour length. In subsequent iterations, a new population of ants is initialized, again randomly placed in different cities. However, this time, their decisions on which city to visit next are informed by both the distance between cities and the updated pheromone levels [21].

The ants continue to construct tours, evaluate their quality, and update the pheromone trails iteratively until a termination criterion is met, such as reaching a predefined number of iterations or achieving a desired solution quality. By leveraging pheromone trail updates and heuristic information, ACO effectively guides the search process towards high-quality solutions to the TSP [58].

2.3.3 ACO for the Waste Management Problem

The Urban Waste Management Problem serves as an excellent example of a real-world TSP instance, offering a balance between complexity and applicability. While it may not be as intricate as some TSP problems found in the literature, it holds significant practical value for cities and municipalities worldwide. Addressing waste management challenges through the optimization of collection routes can lead to substantial cost savings, reduced environmental impact, and improved efficiency of waste collection services [52, 6].

The Travelling Salesperson Problem has found numerous practical applications in various domains, including the urban waste management problem. Waste management plays a critical role in maintaining the cleanliness and sustainability of urban environments. Despite the importance of this application, research on heuristics for waste management problems, particularly for ACO, has often been limited in scope. Most studies in this area focus on optimizing shorter distances, such as a few streets or small neighbourhoods in a continuous space [5, 42, 50].

Although these studies contribute valuable insights into waste management optimization, there is room for expanding their applicability to more extensive urban areas. By exploring ACO algorithms in the context of discrete points on a nonplanar map of Edinburgh, this research aims to bridge the gap between academic studies and real-world applications.

Chapter 3

Methods

This chapter comprehensively describes the ACO algorithm implementation, problem instances, and experimental design employed in this project. Here, we justify the design decisions made to ensure robustness and reliability in evaluating the algorithm's performance across various TSP instances.

3.1 ACO

3.1.1 Definition of ACO

We can explicitly define how we update the pheromones left by the ants every iteration;

Definition 3.1.1 (Pheremone update rule)**.** Given a pheromone matrix τ, the pheromones level between $city_i$ and $city_j$ is given by:

$$\tau_{ij} \leftarrow \rho \cdot \tau_{ij} + (1-\rho) \cdot \sum_{k}^{m} \Delta\tau_{ij}^{k}$$

where ρ is the evaporation rate and

$$\Delta\tau_{ij}^{k} = \begin{cases} Q \cdot \frac{1}{L_k}, & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour,} \\ \sigma \cdot Q \cdot \frac{1}{L_k}, & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour and it is the best tour found,} \\ 0, & \text{otherwise,} \end{cases}$$

where L_k is the length of the tour of ant k.

In this definition, we have introduced some new parameters Q and σ we will go over the function of these later.

The way that ants construct solutions is given by:

Definition 3.1.2 (Construct tour)**.** Given an ant k, the probabilty of *"moving"* from $city_i$ and $city_j$ is given by:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{k}^{m} \tau_{ik}^{\alpha} \cdot \eta_{ik}^{\beta}}, & \text{if j is not already visited} \\ 0, & \text{otherwise} \end{cases}$$

where α is the pheromones bias, β is the heuristic bias and η_{ij} is the heuristic information.

Remark. Mostly $\eta_{ij} = \frac{1}{d_{ij}}$, where d_{ij} is the distance between $city_i$ and $city_j$.

3.1.2 ACO Parameters

At this moment, we understand the main functions that drive ACO. However, we have introduced many parameters, we can now walk through these parameters and how we expect them to behave.

- ρ is the pheromone evaporation rate. This value directly affects how long the colony retains information and how important new information is in relation to previous. A low evaporation rate (ρ close to 1) means that the search strategy will likely be based on exploitation, as most of the knowledge of the previous iterations is kept. Similarly, a high evaporation rate (ρ close to 0) means that the search strategy will likely be based on exploration, where when $\rho = 0$ ants will only have the information of the last iteration on the pheromone matrix.
- m is the number of ants, for all experiments, this will be 20. Literature in ACO shows how this is a near-optimal value for the algorithm [25], and fixing m allows us to better focus on other parameters.
- Q is often used to scale the pheromones levels. However, as all instances of TSP used in the experiments are normalized, Q will remain 1 in all experiments.
- σ is the innovation reward parameter, typically $\sigma = m$. This σ is useful to ensure that no innovation goes unnoticed, no matter how small the improvement, the information stays on the pheromones trails. However, for most experiments this value will also be set to 1, to allow us to focus on more critical parameters such as ρ.
- α is the pheromone bias. Higher values of α produce the search to be more information-focused, ants tend to follow paths that other ants used previously with good results. A lower value of α means that the search is more random or *greedy* depending on β, but the search does not focus as much on previous knowledge.
- β is the heuristic bias. A high β makes the search more *greedy*, as ants follow paths that *seem* short; this makes ants more susceptible to a deceptive or noisy η. A low value of β allows for more exploration, this is important when trying to avoid local optima.
- η is the heuristic information. As explained before, for most experiments this will be the inverse of the distance between the cities. However, on some occasions, we will use a more *noisy* version of this to test the robustness of the algorithm.
- *Max iterations*, some of the decisions in the experiments on how many iterations to wait might seem arbitrary, however, these are mainly made on the basis of "Is the average best solution at that point *good enough*?".

Now that we understand how ACO works and all the parameters around an instance of ACO, let us go over the implementation in Java.

3.2 Java Implementation

The primary goal of the implementation is to provide a well-structured, efficient, and easy-to-understand implementation that facilitates the reproducibility of experiments, the development of new test cases, and the storage of results. This approach not only promotes transparency and reproducibility but also encourages collaboration and knowledge sharing within the research community. One key aspect of this implementation is the use of multithreading, which accelerates the testing process by running several instances of the ACO algorithm concurrently, each with different parameters.

3.2.1 Test Files

To define an experiment or test to run, we can use test files. These are `.txt` files with the following format:

```
nCopies,1
maxIterations,100
nAnts,20
alphas,0,1,101
betas,0,1,101
Qs,1
rhos,0.8
sigmas,1
map_file,normal_att48.txt
output_file,normal_att48_output.csv
```

In this file, we are defining an experiment where we create 1 copy of 10,201 instances of ACO. All of them will run for 100 iterations, have 20 ants, $Q = 1$, $\rho = 0.8$ and $\sigma = 1$. The values of α and β are all possible pairs of the lists from 0 to 1 in 101 steps. Finally, the TSP instance that we are using is `normal_att48.txt` and we output the results on `normal_att48_output.csv`.

3.2.2 The `Main` and `ACO` Classes

The `Main` class parses the test file, which takes as an argument, and creates all the necessary instances of ACO.

The `ACO` class represents an instance of ACO or colony. When an instance of the `ACO` class is created, all ACO parameters are passed from `Main`, such as ρ or the maximum number of iterations. This class contains a `run` function which iterates through the maximum number of iterations, updating ants' trails and pheromones, as described in Definitions 3.1.1 and 3.1.2. It also implements the `Runnable` interface, which allows it to run in multiple threads of a CPU [1].

The key code of this implementation is available in Appendix A.3.

3.3 TSP Problems

We can now discuss the specific TSP problem instances used in this dissertation. This first choice is the widely recognized and well-studied problem instance, the att48 TSP map (see Figure 3.1), representing a 48-city problem instance derived from a real-world scenario. The att48 instance has been extensively used in previous studies, making it ideal for studying the performance of ACO.

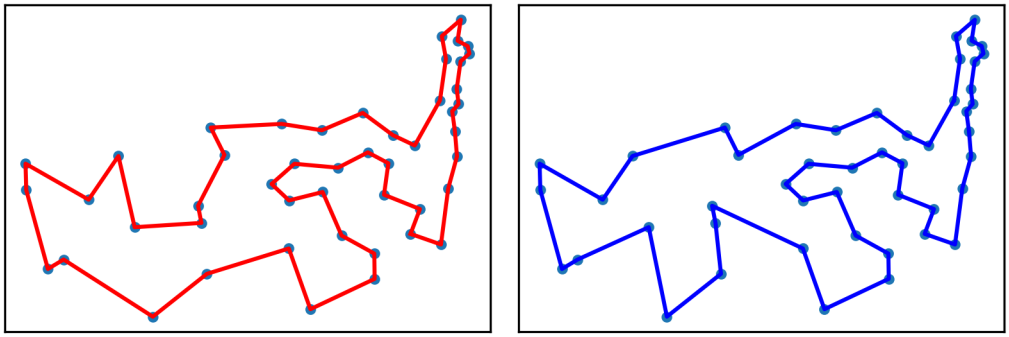

Figure 3.1: Optimal (a) and ACO (b) solutions for att48, with lengths 4.31 and 4.36 respectively. ACO with 1000 iterations, 20 ants, $\rho = 0.9$, $\alpha = 1$, $\beta = 2$, $Q = 1$, $\sigma = 1$.

However, to better understand the behaviour of ACO in various scenarios, we use a normalized version of the att48 problem. In this normalized version, all points are scaled to lie within the unit square. This normalization allows for a more straightforward comparison of results across different problem instances and simplifies the implementation of ACO since there is no need to scale pheromones or heuristics.

In addition to the att48 problem and its normalized version, we consider three other TSP problems, generated by randomly placing points within the unit square. These instances consist of 50, 250, and 1000 points, respectively. The rationale behind selecting these instances is to examine the scalability of the ACO algorithm and its ability to handle problems of varying sizes and complexities. Moreover, using random points within the unit square ensures that the problem instances are not biased towards any particular configuration and can provide a more robust assessment of the algorithm's performance.

3.4 TSP for Edinburgh Communal Bin Collection

The City of Edinburgh presents an excellent example of a nonplanar instance of the Travelling Salesperson Problem. Its streets, characterized by their historical layout and hilly terrain, pose unique challenges for optimization algorithms. In this project, we use the locations of all communal food waste bins in the City Centre of Edinburgh as a highly nonplanar TSP instance. Additionally, we consider all communal clear glass bins in the City as a larger but simpler tour.

In solving these problems, we explore two different heuristic approaches: using the nonplanar real distances between bins and using Euclidean distances.

The data on the bin locations were extracted from The City of Edinburgh Council's website, while the maps were sourced from OpenStreetMap. With this information, we can utilize Python to generate a TSP instance in the form of an adjacency matrix. This process involves calculating the distances between bins and organizing them into a structured format that can be efficiently processed by the ACO algorithm. To find the fastest way to get from one bin to another we use the A* algorithm [20, 16]. The instances of TSP are stored as `.txt` files. The Python code used for this data processing and TSP instance generation is available in Appendix A.2.

By applying the ACO algorithm to these distinct TSP instances, we can gain insights into the performance and adaptability of the algorithm for various problem sizes and complexities. This analysis will help us understand the potential benefits and limitations of using ACO for solving nonplanar instances of TSP, particularly in the context of urban waste management.

Chapter 4

Results

In this chapter, we explore the outcomes of our experiments using ACO for various TSP instances. This chapter aims to provide a comprehensive exploration of the impact of different ACO parameters on the algorithm's performance, efficiency, and solution quality. Through the analysis of various plots and data visualizations, we will draw insights into the relationships between the parameters and the effectiveness of the ACO algorithm in solving TSP problems. Additionally, we will examine how the insights gathered from these analyses can be applied to real-world scenarios, demonstrating the practical implications of optimizing the ACO algorithm parameters for solving complex TSP instances and showcasing the algorithm's potential in addressing real-life challenges.

4.1 The Effects of Pheromones and Heuristic Bias

The parameters α and β, pheromones and heuristic bias respectively, play a crucial role in the ACO algorithm by determining the relative importance of pheromone trails and heuristic information, given by Formula 3.1.2. α controls the influence of the pheromone trail, while β controls the influence of the heuristic information (e.g., the distance between two cities). By adjusting these parameters, we can modulate the balance between exploration and exploitation in the ACO algorithm, ultimately affecting its performance.

To examine the effects of varying α and β on the ACO algorithm's performance, we conduct a series of experiments using different combinations of these parameters. The results are visualized as filled contour plots, where the y-axis represents the α values, the x-axis represents the β values, and the colour indicates the quality of the solution found (redder colours indicate better solutions).

It is generally expected that a lower α value would promote exploration, as ants would be less inclined to follow existing pheromone trails and more likely to discover new paths. Conversely, a higher α value would encourage exploitation, with ants more likely to follow the stronger pheromone trails, potentially leading to faster convergence but a higher risk of getting trapped in local optima.

On the other hand, a lower β value would cause ants to rely less on the heuristic information, potentially making their search more random and less directed towards shorter paths. Meanwhile, a higher β value would emphasize the heuristic information, guiding ants towards shorter paths more effectively but possibly causing premature convergence.

Let us now look at Figure 4.1. Here, we can see how after 10 iterations of ACO when $\alpha = 2$ and $\beta > 2$, seem to yield the best performance, striking a balance between exploration and exploitation. Another important remark is how the value of α is critical to the performance of the algorithm while β is a lot more forgiving.

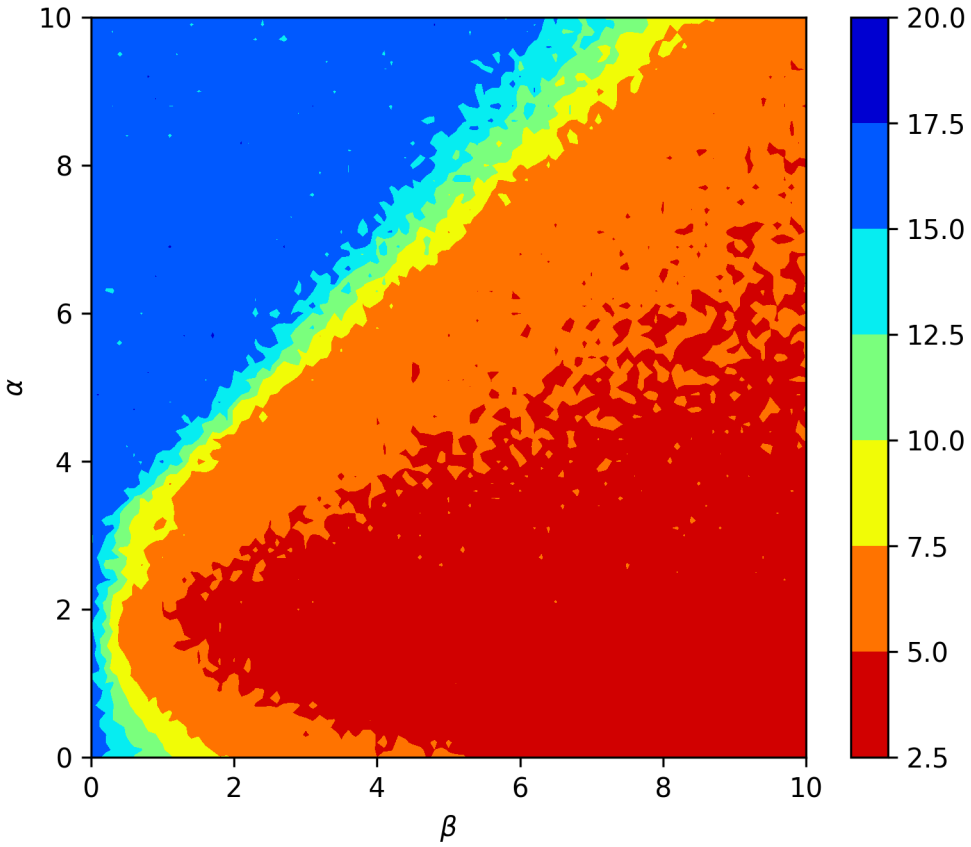

Figure 4.1: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 10 iterations on normalised att48. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

Now, we can look at the same experiment stopping after 5, 10 and 50 iterations. From Figure 4.2, we can see that:

- As iterations increase, higher values of α became acceptable. As ACO needs more iterations for a more exploitation-based search to yield good results.
- As iterations increase, the critical point of α that splits *good* and *bad* solutions seems to approach 1.

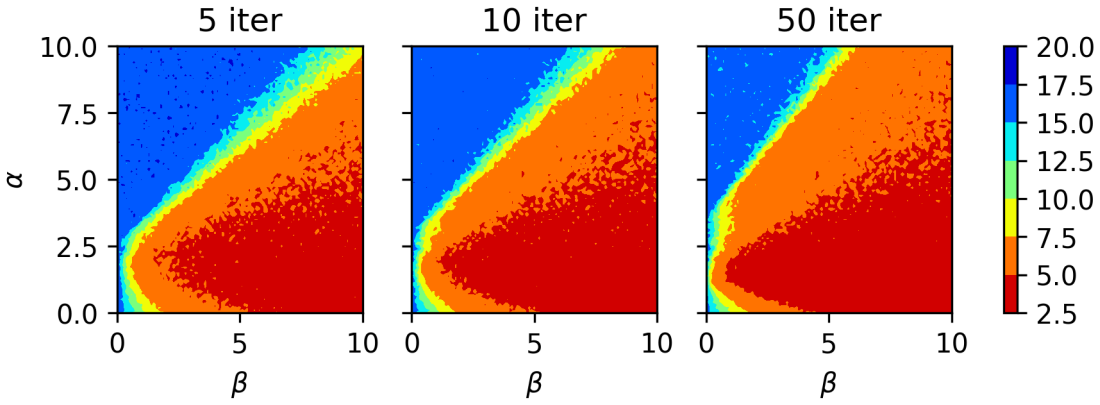

Figure 4.2: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 5 10 and 50 iterations on normalised att48. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

After looking at these results for att48, we might question if this behaviour is consistent with a randomly generated map. We can see similar results in Figures 4.3 and 4.4. Here instate of using att48 as the map for TSP, we use a randomly generated map with 50 cities. The only recognisable difference between the results for these two maps is the decrease of noise in the plots and the lines that split the lines are straighter, likely because att48 is more deceptive than 50 random points.

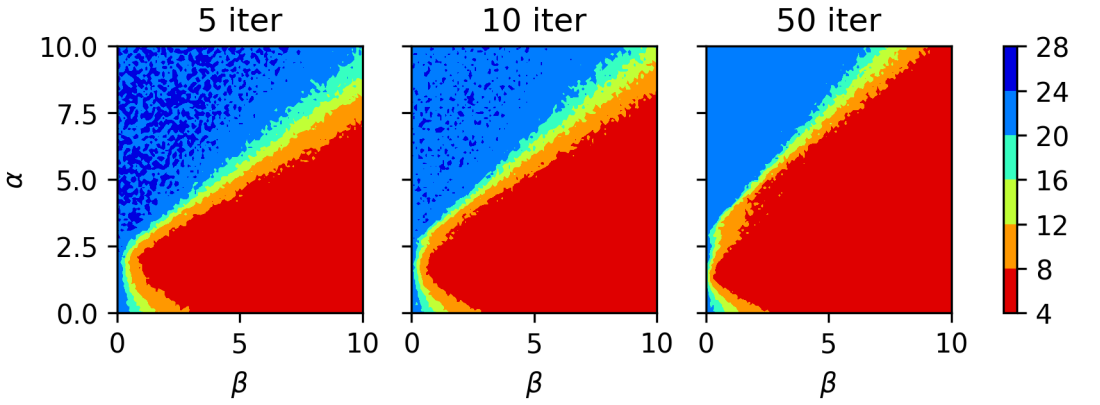

Figure 4.3: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 5 10 and 50 iterations on 50 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

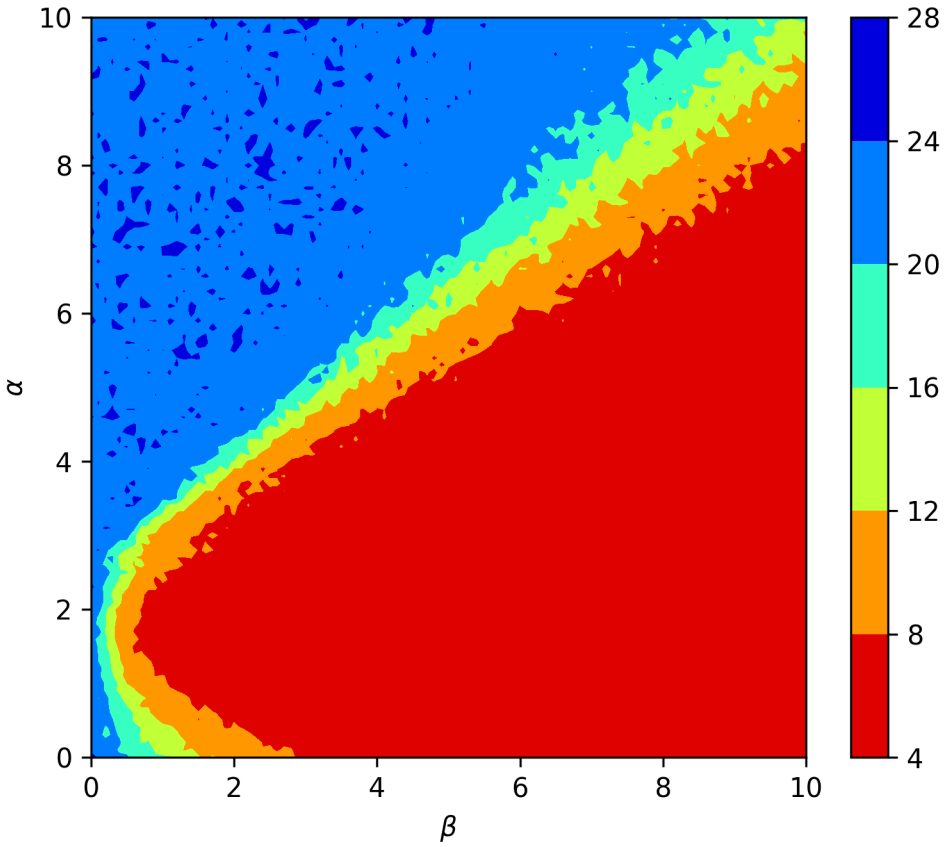

Figure 4.4: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 10 iterations on 50 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

At this point, we may question if this parameter space of $\alpha, \beta \in (0, 10)$ is sufficient.

In Figure 4.5 we can see an exploration of the negative values of α and β. The results are in line with our expectations, for negative α values there is no exploitation and all exploration is only led by the heuristic information. For negative values of β ACO does not perform at all, this is very logical since ants would be attracted to choose paths with a longer distance between cities.

All the plots in this section are produced in small instances of TSP instead of bigger ones. This is because most of the tests done in this study are done in a smaller amount of cities to allow for a faster runtime and smoother plots, when the size of the TSP map does not affect the results. This is the case for all the tests in this section, we can see in Appendix B how results for 250 cities look very similar.

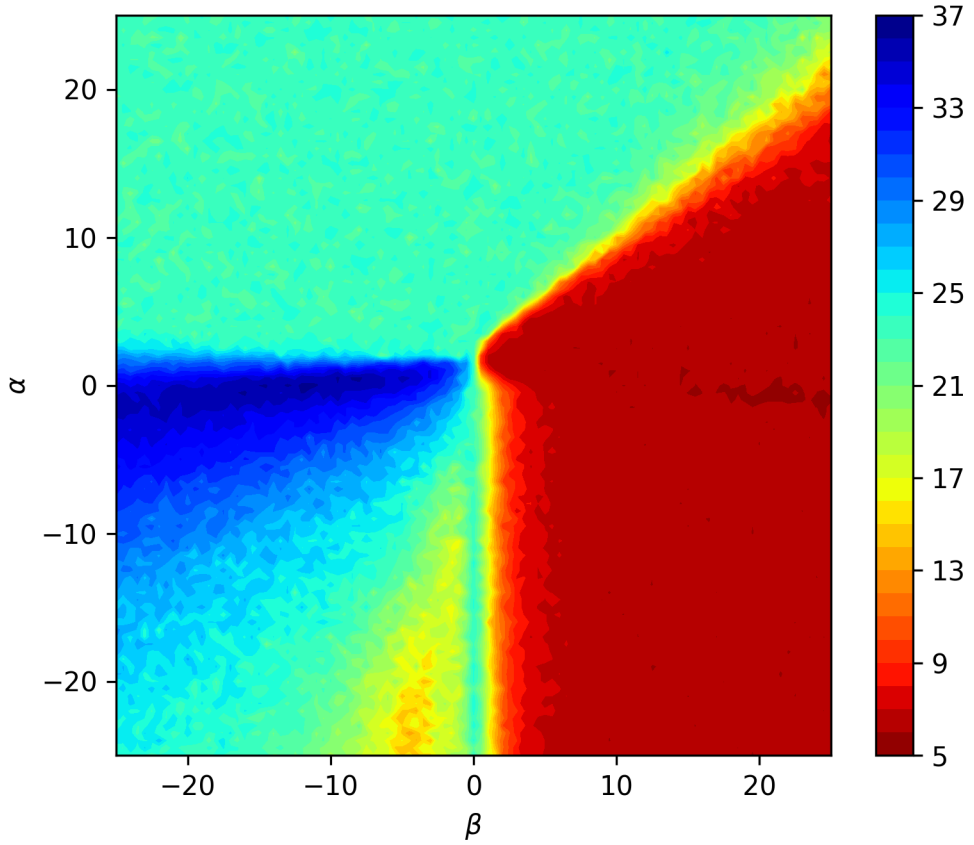

Figure 4.5: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (-25, 25)$, after 10 iterations on 50 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

4.2 The Effects of a Noisy Heuristic

It is also interesting to see the effect of a noisy distance function on the α and β filled contour plots.

In Figure 4.6 we use a noisy heuristic function η', which is defined by:

Definition 4.2.1 (Noisy heuristic function η')**.**

$$\eta'_{ij} = \frac{1}{2 \cdot r \cdot d_{ij}},$$

where d_{ij} is the distance between $city_i$ and $city_j$ and r is a uniformly random number from 0 to 1.

Remarkably, Fig. 4.6 illustrates that there is no optimal value of α for which β can change and the algorithm can still yield good results. Their relationship seems to be similar to a one-to-one correspondence of α and β which maintains the best results.

When compared to Figures 4.1 and 4.4, $\alpha = \beta = 2$ looks like the only common place yielding good solutions.

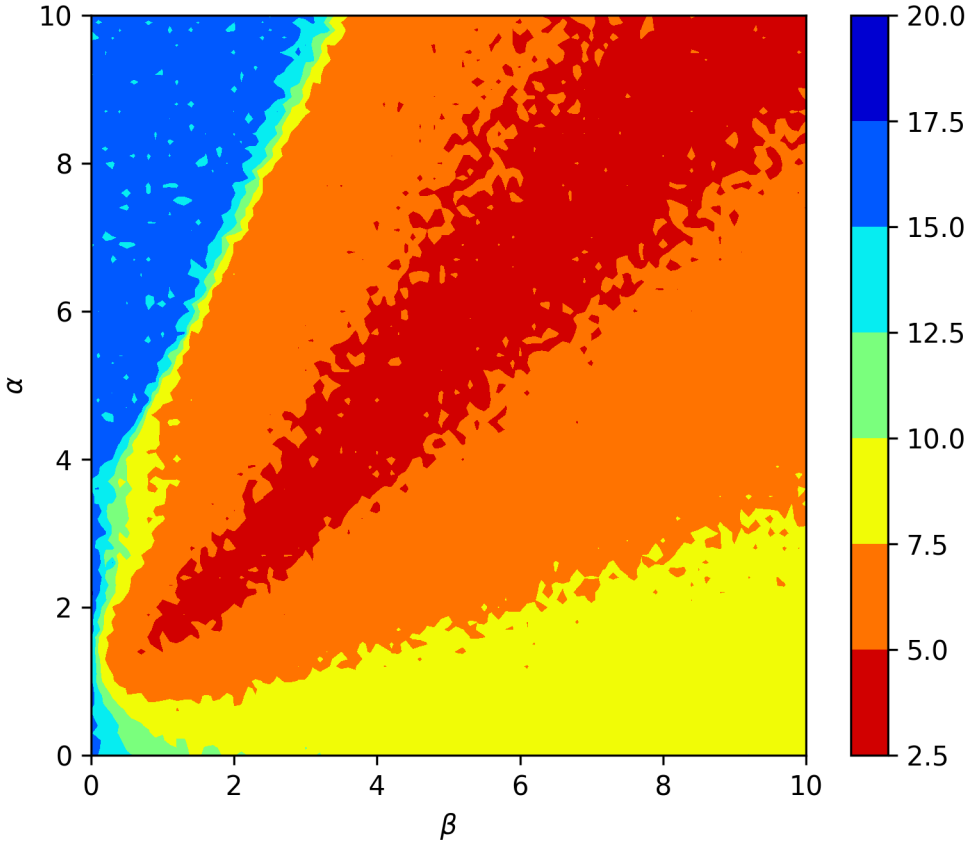

Figure 4.6: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 50 iterations on normalised att48 with a noisy heuristic function. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

4.3 The Effects of an Innovation Reward

The parameter σ serves as an innovation reward as shown in Definition 3.1.1. Figure 4.7 depicts a comparison of $\sigma = 1$ and $\sigma = 20$ ACO instances, which remarkably illustrated the disappearance of the critical line between good and bad solutions. The instances where $\sigma = 20$ find the best solutions, as can be seem by the darker shade of in Figure 4.7 that does not appear when $\sigma = 1$.

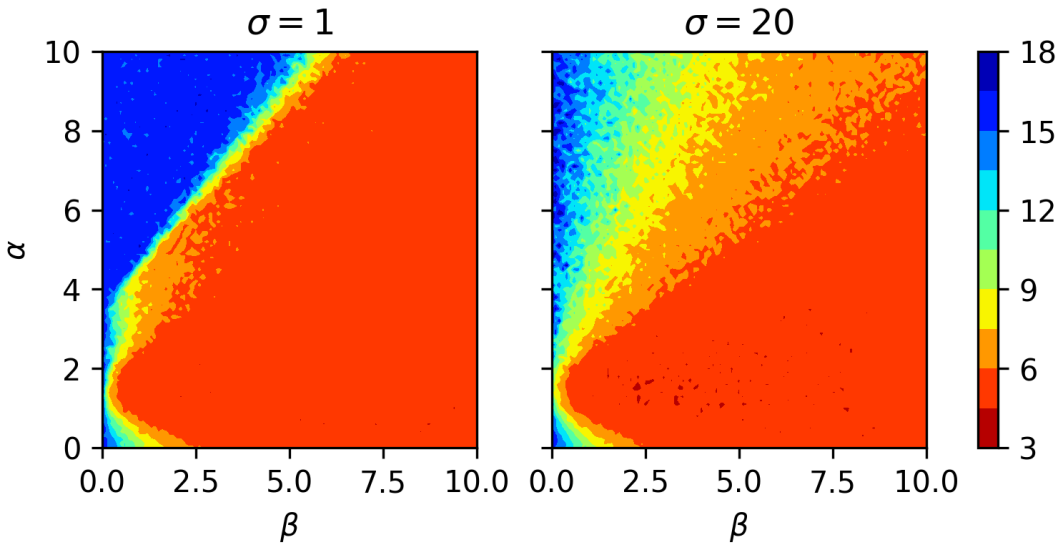

Figure 4.7: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$ and $\sigma = 1$ or $\sigma = 20$, after 50 iterations on 50 random cities. Other parameter values are $\rho = 0.8$, $m = 20$ and $Q = 1$.

4.4 The Effects of the Evaporation Rate

The evaporation rate ρ is a key parameter in the ACO algorithm that governs the rate at which pheromone trails decay over time. This parameter plays a crucial role in balancing the exploration and exploitation of the search space by the algorithm. By adjusting the evaporation rate, we can influence the algorithm's ability to escape local optima and converge to global optima.

A high evaporation rate (ρ closer to 0) causes the pheromone trails to decay rapidly, promoting exploration and allowing the algorithm to escape local optima more easily. However, this can also lead to slower convergence, as the pheromone trails do not persist long enough to guide the ants effectively towards better solutions.

On the other hand, a low evaporation rate (ρ closer to 1) results in slower decay of the pheromone trails, encouraging exploitation and potentially leading to faster convergence. However, this increased exploitation can also increase the likelihood of the algorithm getting trapped in local optima. The ants may become overly focused on a particular area of the search space and less likely to explore new paths.

To investigate the effects of varying ρ on the ACO algorithm's performance, we can experiment using different evaporation rates while keeping the other parameters constant. Based on the results from the previous experiments, let us fix $\alpha = 1$ and $\beta = 2$. Now we can run 100 instances of ACO with different values of $\rho \in (0, 1)$. If we stop at iteration i, and plot the results as ρ against the length of the best path, we can see the value of ρ for which ACO performs best. The specific method to find this value is accessible in Appendix A.2. We can see in Figure 4.8; for lower iterations, lower values of ρ perform best; for higher iterations, higher values of ρ perform best. This is already an interesting phenomenon, indicating a clear trend tend on the values of ρ.

Now, that we see this trend, we can plot this *best value of* ρ for each iteration, as shown in Figure 4.9. Here, we can see this functional dependency of ρ and the maximum number of iterations. We will explore this further in the next chapter.

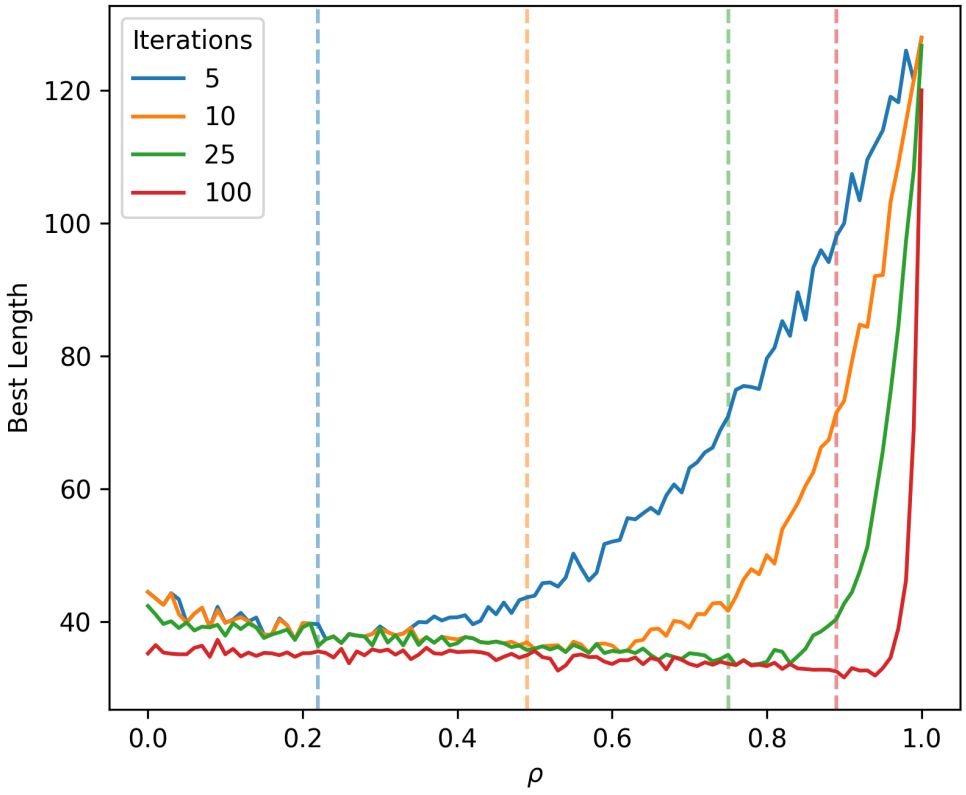

Figure 4.8: Line plots showing the best length for each value of $\rho \in (0, 1)$ for 5, 10, 25 and 100 iterations. Vertical lines show the midpoint of the lowest average area of the line, this is the "best value of ρ for i iterations". The problem instance is 1000 random cities, other parameter values are $\alpha = 1$, $\beta = 2$, $m = 20$, $Q = 1$ and $\sigma = 1$.

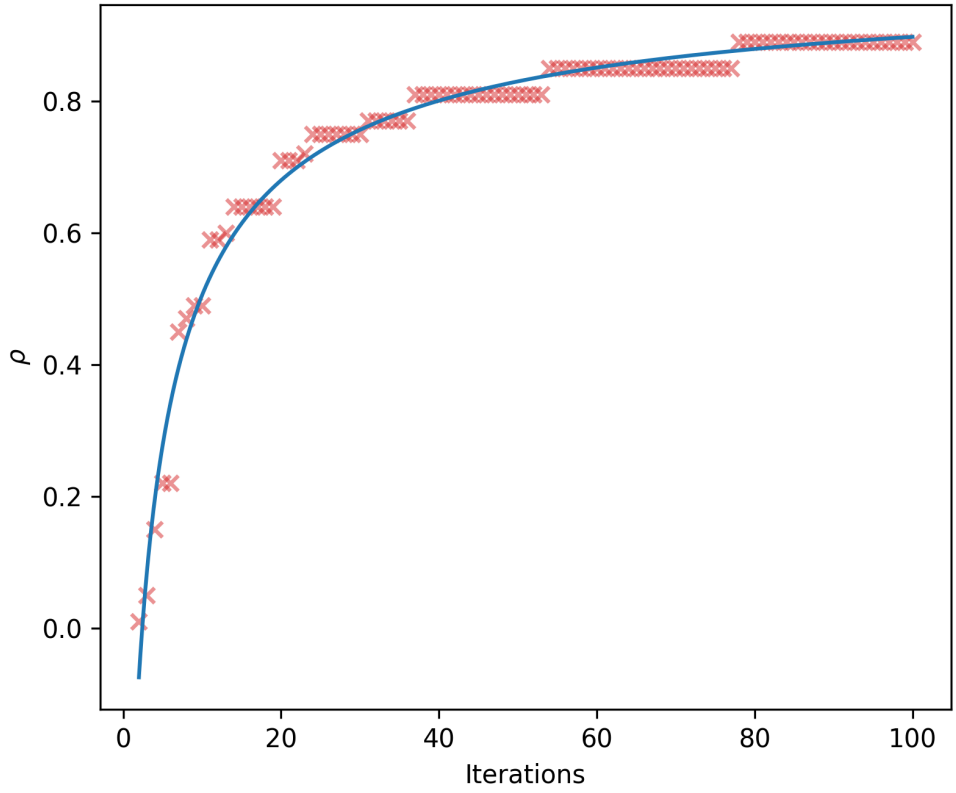

Figure 4.9: Plot ”best value of ρ” for each iteration. The red markers are the ”best value of ρ” found for each iteration. The blue line is the function $1 - \frac{4}{x^{0.784} + 2}$ plot on top to estimate the behaviour of the ”best value of ρ”. The problem instance is 1000 random cities, other parameter values are $\alpha = 1$, $\beta = 2$, $m = 20$, $Q = 1$ and $\sigma = 1$.

4.5 Real-world Experiments

Let us look at the results of implementing ACO for Urban Waste Management. The first instance of this problem is all communal clear glass bins in Edinburgh. For the algorithm the is no real difference in approaching this practical problem or a theoretical one, though the construction of the problem statement and the choice of heuristics varies.

Figure 4.10 shows the use of the Euclidean distance between bins to construct the problem. The calculation of this straight distance between the bins is based on their actual geographic coordinates.

In 4.11 we first construct the adjacency matrix of the problem statement with the real distance on the streets of Edinburgh as explained in Subsection 3.4. The paths between these two approaches are similar and so is the tour length, 73.4km and 71.1km, respectively. While the results do not differ greatly, the approach applying a non-planar heuristic to solve this non-planar problem produces a more desirable outcome and is favoured. This is not particularly surprising, as the bin locations form a route which is not deceptive when using a Euclidean heuristic.

Perhaps more noteworthy are the results of the second experiment in which we look at the city centre of Edinburgh instead of the whole city. The city centre of Edinburgh is exceptionally nonplanar and deceptive; with tunnels, one-directional streets, bridges and dead-end roads. The application of a non-planar heuristic to this real-world structure significantly improves the algorithm's solution from 37.9 km to 27.2 km, visible in Figures 4.12 and 4.13

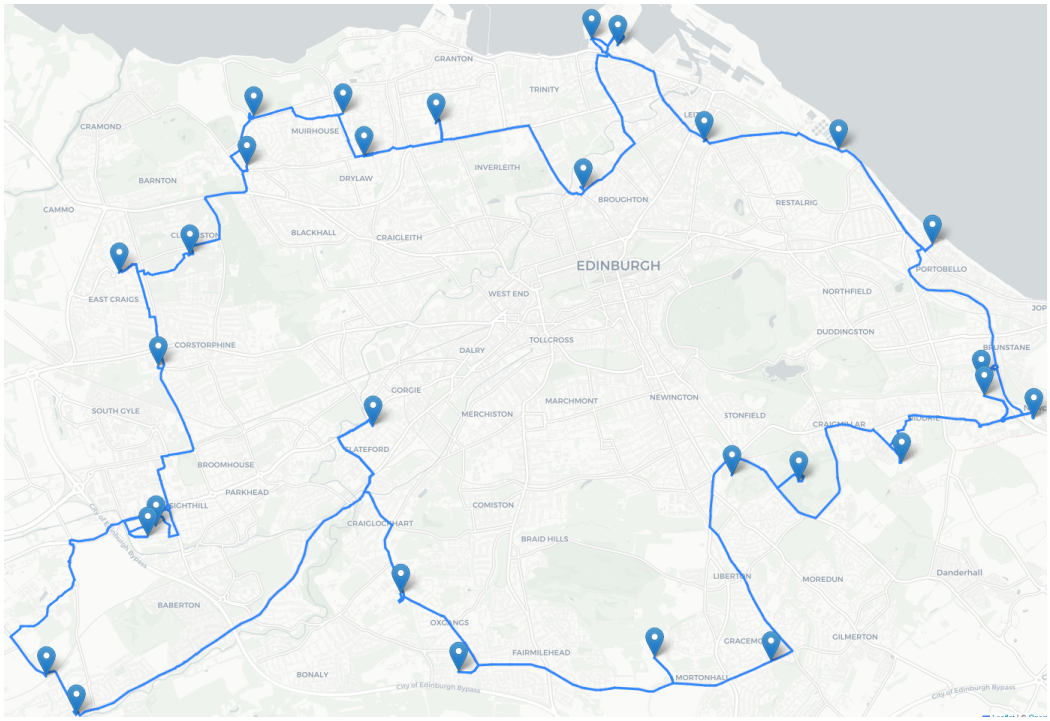

Figure 4.10: Communal clear glass bins in Edinburgh, size 29, 1000 iterations, $\alpha = 1$, $\beta = 2$, $\rho = 0.8$, $Q = 1$, $\sigma = 1$, 20 ants, heuristic Euclidean distance. The tour length is 73.4 km.

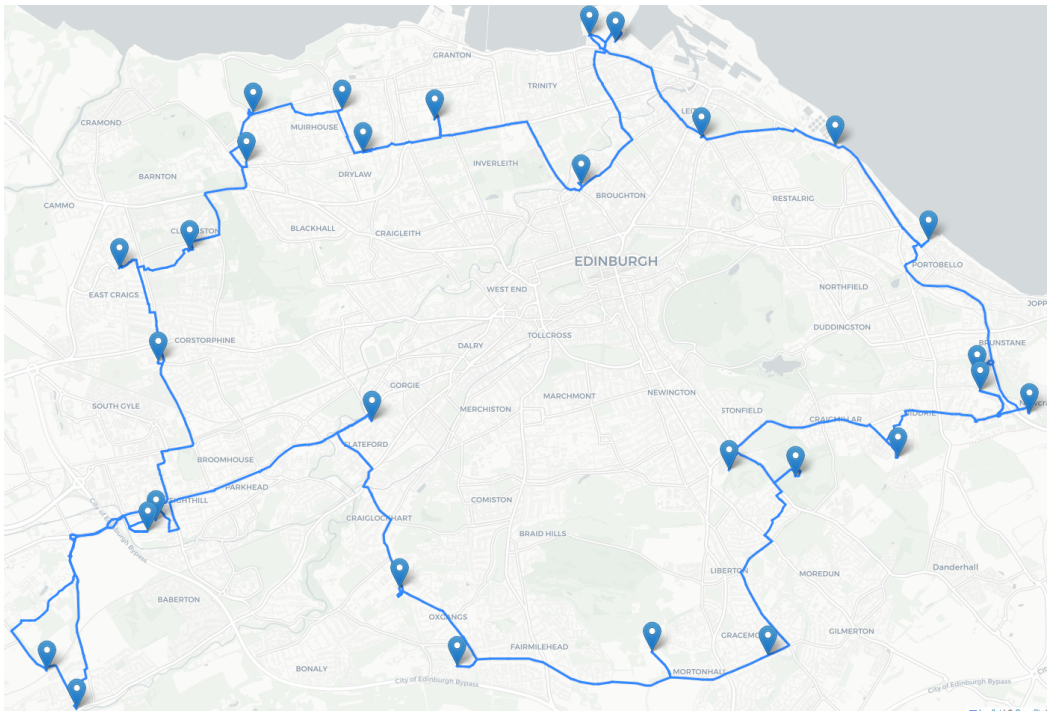

Figure 4.11: Communal clear glass bins in Edinburgh, size 29, 1000 iterations, $\alpha = 1$, $\beta = 2$, $\rho = 0.8$, $Q = 1$, $\sigma = 1$, 20 ants, heuristic real distance (non-planar). The tour length is 71.1 km.

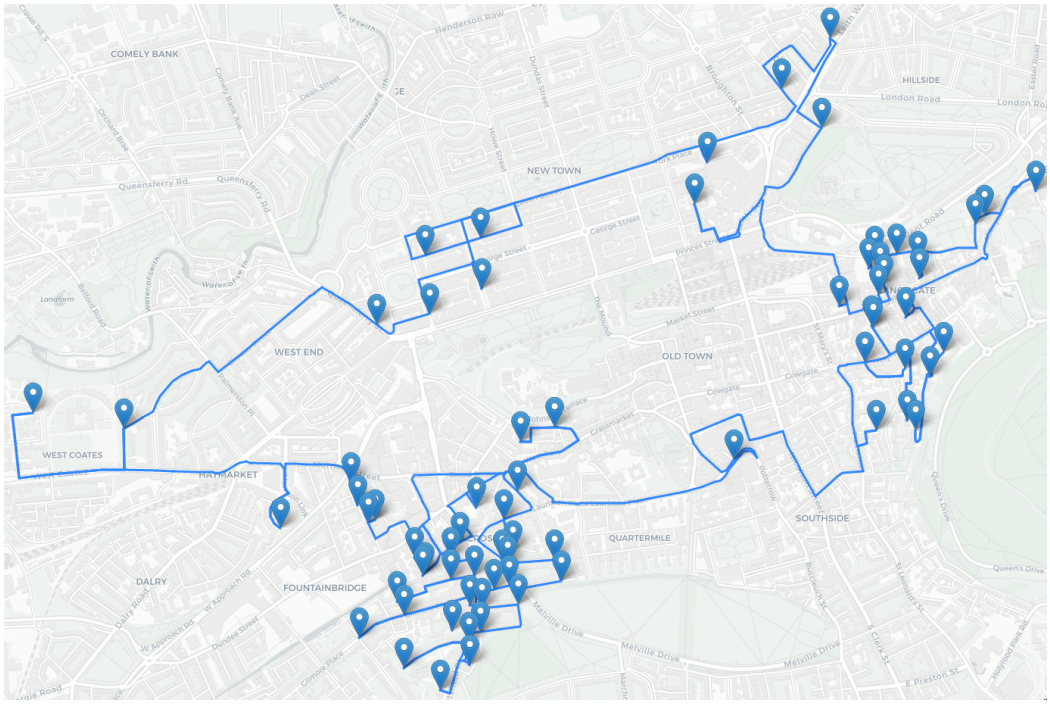

Figure 4.12: Communal food waste bins in the City Centre of Edinburgh, size 71, 1000 iterations, $\alpha = 1$, $\beta = 2$, $\rho = 0.8$, $Q = 1$, $\sigma = 1$, 20 ants, heuristic real distance (non-planar). The tour length is 27.2 km.

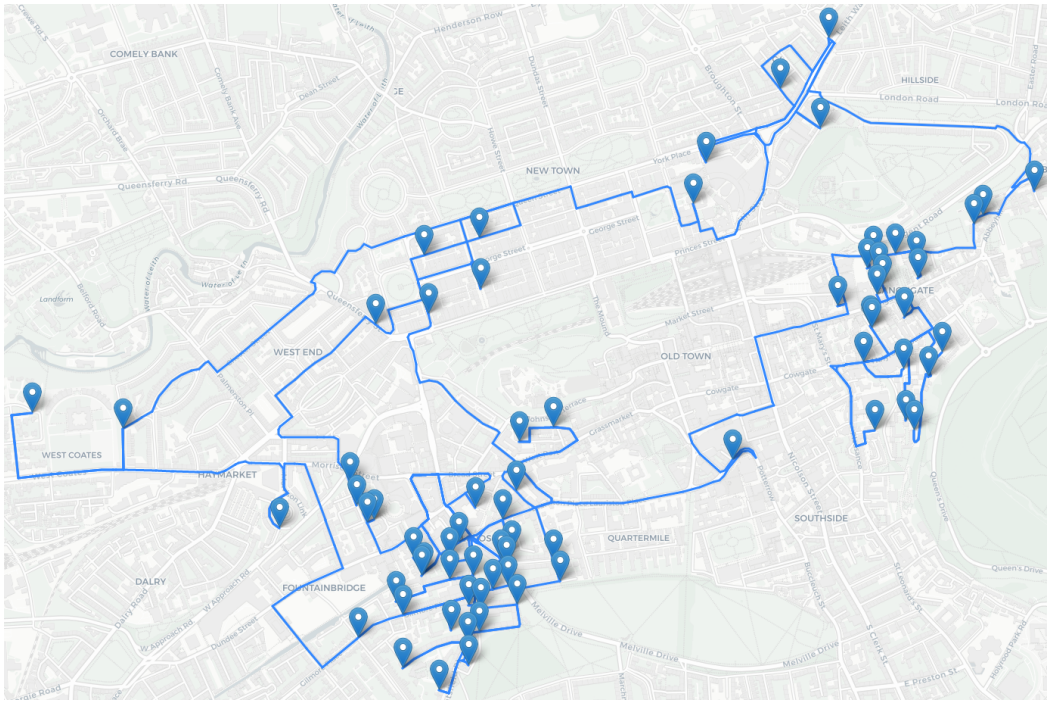

Figure 4.13: Communal food waste bins in the City Centre of Edinburgh, size 71, 1000 iterations, $\alpha = 1$, $\beta = 2$, $\rho = 0.8$, $Q = 1$, $\sigma = 1$, 20 ants, heuristic Euclidean distance. The tour length is 37.9 km.

Chapter 5

Discussion

The results of these experiments on the ACO algorithm for solving TSP provide valuable insights into the impact of different ACO parameters on the algorithm's performance and solution quality. We will explore these insights in this chapter.

5.1 The Role of Metaheuristics

Metaheuristic optimization techniques have been instrumental in addressing complex optimization problems that are often intractable using conventional methods. By intelligently exploring the search space and exploiting accumulated knowledge, metaheuristics deliver efficient and effective solutions to a broad range of optimization challenges. However, it is crucial to maintain a critical perspective on their development process and ensure that scientific rigour is preserved.

In his paper "Metaheuristics—the metaphor exposed," Kenneth Sörensen criticizes the overreliance on natural or man-made processes as a source of inspiration for new optimization algorithms [55]. According to Sörensen, the proliferation of metaphor-based metaheuristics has led the field away from scientific rigour and resulted in the development of algorithms that may not offer significant improvements over existing methods. He contends that the extensive use of metaphors as inspiration and justification for new methods has created a vulnerability in the metaheuristics field, making it susceptible to unscientific practices and fallacies.

Despite these concerns, it is necessary to acknowledge that innovative and high-quality research is still being conducted within the field of metaheuristics. The key to maintaining scientific rigour lies in thoroughly understanding the underlying principles that drive the success of natural systems, rather than merely replicating their superficial characteristics. Moreover, the development of new metaheuristics should be supported by rigorous empirical and theoretical analysis to demonstrate their effectiveness, efficiency, and scalability across various problem domains.

Given the points raised by Sörensen, the discussion in this dissertation will attempt to move away from heavy reliance on the ant metaphor and instead focus on the parameter space and performance of the ACO algorithm. By conducting extensive experiments

and analyzing the impact of different parameter settings on ACO's performance, researchers can gain a deeper understanding of the algorithm's behaviour, leading to a more informed and efficient application of the algorithm in solving optimization problems.

While it is vital to remain critical of the overreliance on nature-inspired metaphors when developing metaheuristics, it is undeniable that some of these methods have made significant contributions to the field of optimization, as discussed in the Background chapter. The challenge for researchers is to strike a balance between drawing inspiration from natural systems and maintaining scientific rigour by thoroughly understanding the principles behind these systems and evaluating the performance of the resulting algorithms. The following discussion aims to contribute to achieving this balance, providing valuable insights into the parameter space and performance of ACO.

5.2 The Role of the Exponents α and β

Our experimental results offer valuable insights into the interplay between the pheromones and heuristic bias in the ACO algorithm. Through a series of experiments, we have observed the impact of varying α and β values on the algorithm's performance in solving the TSP. Our findings highlight the importance of striking a delicate balance between exploration and exploitation for achieving optimal performance.

The filled contour plots serve as a visual representation of the quality of solutions obtained for different combinations of α and β values. As hypothesized, lower α values promote exploration by reducing the reliance on pheromone trails, while higher α values encourage exploitation by increasing the influence of pheromone trails. Similarly, lower β values diminish the impact of heuristic information, making the search process more random, while higher β values accentuate the heuristic information, guiding ants more effectively towards shorter paths.

The results indicate that the best performance is achieved when $\alpha = 2$ and $\beta > 2$. This balance between exploration and exploitation allows the algorithm to efficiently search the solution space, while still benefiting from the accumulated knowledge of the ants. The fact that the value of α has a more significant impact on the algorithm's performance than the value of β might be attributed to the critical role that pheromones play in guiding ants' decisions. This observation is consistent across different TSP instances, including both the att48 and randomly generated maps with 50 cities.

Furthermore, our findings suggest that as the number of iterations increases, higher values of α become more acceptable. This could be because a more exploitation-based search requires more iterations to yield good results. The critical point of α that separates good and bad solutions appears to approach 1 as the number of iterations increases, which may indicate the algorithm's adaptability to different stages of the search process.

Our exploration of the effects of a noisy heuristic function on the performance of the ACO algorithm is particularly interesting. The noisy heuristic function introduces uncertainty in the distance information, which in turn affects the algorithm's decision-

making process. In the presence of a noisy heuristic function, there is no single optimal value of α that can yield good results for all values of β. Instead, we observe a more complex relationship between α and β, with a one-to-one correspondence between the two parameters that maintain the best results. This finding emphasizes the importance of considering the quality and reliability of heuristic information when tuning the ACO algorithm's parameters.

Perhaps, some of the most surprising results are the ones shown in Figure 4.5, which seems like a negative α can still produce good performance on ACO. However, we should think about how these results are oblation, when α is negative ants are attracted to paths with smaller pheromone levels. This means that these paths could not be obtained by exploitation of good paths since the information that the pheromone matrix holds is not accessible to ants for negative α. We can now hypnotise, these paths that seem to be as good as the paths when $\alpha, \beta > 0$ must have been found through random and greedy search.

To test this we can use the length of the *"current best path"* instated of the *"best path found"*. ACO can often lose information if the parameter values are not appropriate. When we subtract the length of the *"current best path"* to the length *"best path found"*, we get an estimated value for how much information are we retaining, where the closer to zero the more information about the best path we are keeping on the pheromones trails. The result of this is Figure 5.1, here we can clearly see, negative α values although originally seemed to perform adequately, only produce results randomly and through greedy search and should be avoided, particularly when working with larger and more deceptive instances of TSP.

5.3 The Role of the Innovation Reward σ

In this section, we evaluate the impact of the innovation reward, controlled by the parameter σ, on the performance of the ACO algorithm when solving the TSP. The comparison of the quality of solutions obtained for different combinations of α and β values when $\sigma = 1$ and $\sigma = 20$, as shown in Figure 4.7, offers interesting insights into the role of innovation rewards in the ACO algorithm.

The disappearance of the critical line separating good and bad solutions, previously observed in experiments without the innovation reward, suggests that a higher innovation reward ($\sigma = 20$) may help the algorithm explore the solution space more effectively. The increased exploration of new paths can potentially enable the ACO algorithm to avoid local optima, ultimately leading to better solutions.

Furthermore, the darker shade of red in the plot for instances with $\sigma = 20$ implies that higher innovation rewards result in the discovery of better solutions. It is possible that the higher value of σ encourages ants to explore new paths more effectively, leading to better solutions by escaping from local optima more easily.

It is worth considering the possible reasons for these observations. One hypothesis is that the higher innovation reward promotes a more balanced exploration-exploitation trade-off, allowing the algorithm to search for better solutions while still utilizing

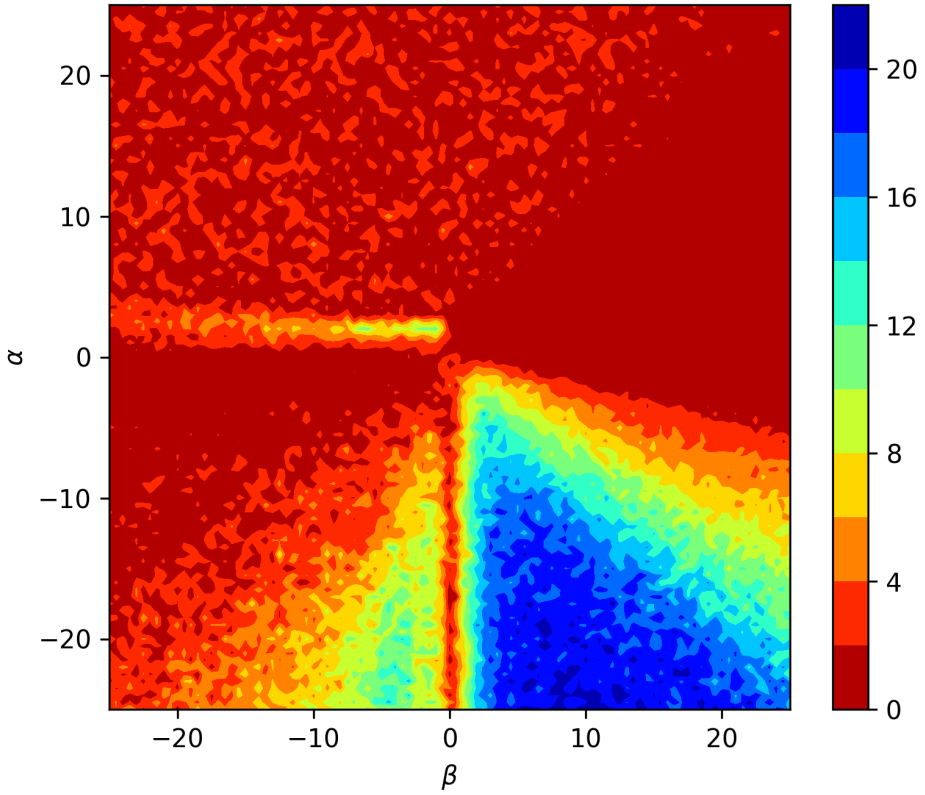

Figure 5.1: Filled contour plot of the *"best current length"−"best length"* of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 10 iterations on normalised 50 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

the accumulated pheromone information. Another hypothesis is that the increased innovation reward may prevent the algorithm from getting trapped in local optima, as ants are heavily rewarded for every small innovation.

However, it is crucial to critically assess the results and question whether the paths generated with $\sigma = 20$ are genuinely better.

So from Figure 4.7 we can take the best instances of both plots and see how the path they describe looks. This is what we find in Figure 5.2, and even though both paths look very similar, the path provided by the instance of ACO where $\sigma = 1$ is a self-intersecting path which is always suboptimal in a Euclidean version of TSP. The path provided by the instance of ACO where $\sigma = 20$ is not a self-intersecting path, and it also has a shorter length.

This already looks promising, $\sigma = 20$ provides better structured and shorter paths. To look at this further, we can run 20,000 copies of ACO for 50 iterations, half of them for $\sigma = 1$ and the other half for $\sigma = 20$. We can see the results of this experiment in Figure 5.3. In this Figure we see two plots, if we first focus on the left plot, we see how for a small number of iterations (< 10), $\sigma = 1$ seems to yield better results on average, this is

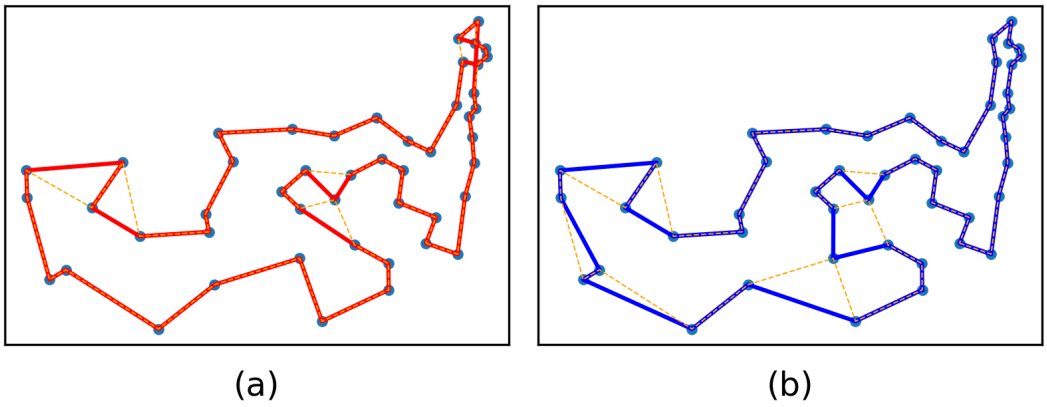

Figure 5.2: Two different routes for att48. (a) is self-intersecting, (b) is not. In discontinuous orange lines the optimal route.

likely because for not many iterations a greedy approach without much interest in the pheromones trails is sufficient. However, as iterations pass, $\sigma = 20$ performs on average better than $\sigma = 1$. If we now look at the right side of Figure 5.3, we find a probability histogram of the *"best length"* found for each instance of ACO after 50 iterations. This histogram shows clearly how $\sigma = 20$ produces better results for ACO. Also, from this histogram we find that $\sigma = 1$ seems to produce a normal distribution when plotted in this manner; however, the distribution of $\sigma = 20$ seems to have a longer tail and favours shorter solutions.

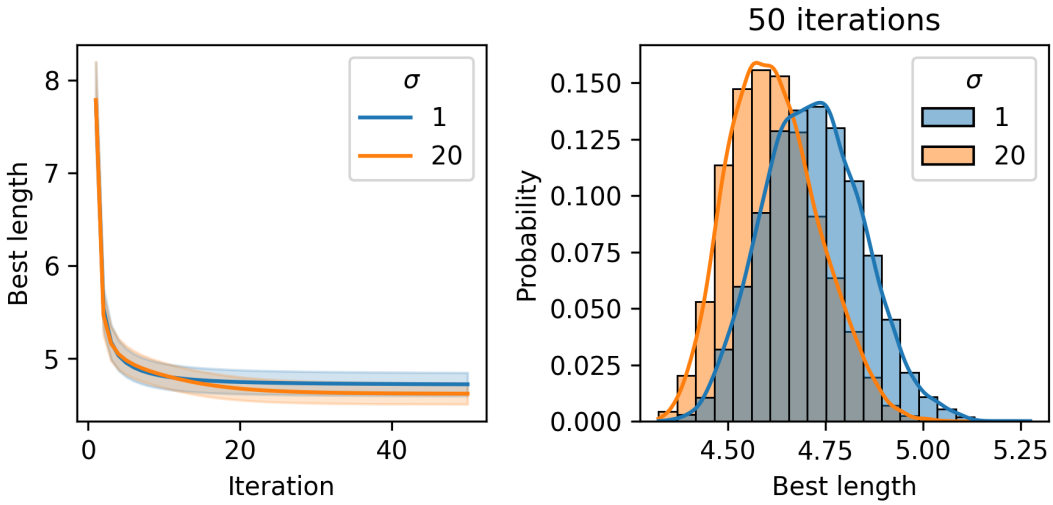

Figure 5.3: Comparison of 20,000 instances of ACO for 50 iterations on 50 random cities, for $\sigma = 1$ and $\sigma = 2$. Other parameter values are $\alpha = 1$, $\beta = 2$, $m = 20$, $Q = 1$ and $\rho = 0.8$.

These results highlight the importance of understanding the impact of varying the σ value on the ACO algorithm's performance and show promise for future research. Further investigation is needed to determine the optimal range of σ values for different

problem instances and search scenarios and to better comprehend the role of innovation rewards in the ACO algorithm.

5.4 The Role of the Evaporation Rate ρ

The evaporation rate, ρ, is a critical parameter in the ACO algorithm that determines the rate at which pheromone trails decay over time. By modifying the evaporation rate, we can affect the algorithm's balance between exploration and exploitation, ultimately influencing its ability to escape local optima and converge to global optima. A high evaporation rate (ρ closer to 0) places greater emphasis on recent information, promoting exploitation by reducing the influence of past information. On the other hand, a low evaporation rate (ρ closer to 1) encourages exploration by allowing pheromone trails to persist, guiding ants towards previously discovered promising paths.

We recall from Definition 3.1.1 that the pheromone update rule incorporates the evaporation rate, ρ. With a high evaporation rate, the algorithm favours more recent information from the ants' tours, making it more adaptive to changes in the search space. This adaptability can be particularly useful when the algorithm has not yet explored most of the search space and should not over-commit to any solutions.

Our experiments have revealed an interesting relationship between the optimal evaporation rate and the number of iterations. The function

$$1 - \frac{4}{i^{0.784} + 2} \tag{5.1}$$

provides an estimation of the optimal evaporation rate, ρ, as a function of the number of iterations, i. As the number of iterations increases, the optimal ρ approaches a higher value, which minimizes the value of recent information and strikes a balance between exploration and exploitation.

One possible explanation for this relationship is that in the early stages of the algorithm, the ants have only explored a small portion of the search space and the knowledge accumulated by the pheromone trails is not reliable or informative. In these early iterations having a high evaporation rate allows the algorithm to avoid overreliance on insufficient pheromone trails, while also giving the algorithm space for adaptability and change.

As the number of iterations increases, the ants have explored a larger portion of the search space, and their accumulated knowledge becomes more reliable. In this context, a lower evaporation rate allows the algorithm to rely more upon pheromones trails and exploit the small variations of already good paths.

It is essential to consider how these findings might generalize to smaller or larger TSP instances. For smaller TSP instances, the search space is less complex, and a greater emphasis on exploration might be beneficial. In such cases, the optimal ρ might converge more slowly to a value further away from 1. Conversely, for larger TSP instances, the search space becomes more complex, and a more balanced approach between exploration and exploitation might be necessary. The optimal value of ρ might

converge to a number closer to 1 as iterations pass at a faster rate, as accumulating information on larger instances of TSP becomes relatively more important than for smaller ones.

Furthermore, the ants' route construction process, as described in Definition 3.1.2, is also influenced by the evaporation rate. A high evaporation rate leads to more volatile pheromone trails, which can affect the ants' decision-making process when choosing the next city to visit. This increased volatility could, in turn, promote exploration and help the ants avoid local optima.

Understanding the relationship between the evaporation rate and the number of iterations is vital for optimizing the ACO algorithm's performance. By adjusting the evaporation rate according to the problem's complexity, the desired balance between exploration and exploitation, and the ants' accumulated knowledge, we can fine-tune the algorithm to perform effectively across various problem sizes and search space complexities. By analyzing the relationship between ρ and the number of iterations and adapting it to different problem instances, we can improve the efficiency and effectiveness of the ACO algorithm in solving TSP and other optimization problems.

An interesting direction for future work is to investigate how the optimal evaporation rate and its relationship with the number of iterations vary across different types of problem instances, such as those with varying degrees of noise or constraints. By doing so, we can develop a more comprehensive understanding of the role of the evaporation rate in the ACO algorithm and potentially devise adaptive strategies for adjusting the evaporation rate in response to the problem's characteristics.

Lastly, it would be valuable to explore the impact of dynamic evaporation rates, where the evaporation rate is adjusted over time according to the algorithm's progress. We can test this briefly; if we take the function that we defined in Function 5.1, we can compare 20 ACO instances, 10 have $\rho = 0.8$ and the other 10 have ρ as defined in Function 5.1. The result of this comparison can be seen in Figure 5.4. This dynamic approach for ρ seems very promising. Looking at both runs, the first thing that we realise is that both lines behave the same for the first two iterations. This is because we can only use the function defined in Function 5.1 when $i > 2$ so that $\rho \in (0,1)$, else we use $\rho = 0.8$. After these two iterations, it is clear that the dynamic approach significantly outperforms $\rho = 0.8$ for the first ≈ 20 iterations. At this point, $\rho = 0.8$ produces better results, as expected based on Figure 4.9, which shows that $\rho = 0.8$ performs best in this range. It also seems that by the time ACO stops, reaching 100 iterations, both variants are performing similarly. However, the dynamic approach has been continuously improving meanwhile the constant approach has not.

This last experiment, even though inconclusive, as it only runs for 100 iterations and has just 10 copies of each instance, shows a very real promise in a dynamic ρ approach. Such a dynamic approach could enable the ACO algorithm to adapt more effectively to various problem instances, further enhancing its performance in solving complex optimization problems.

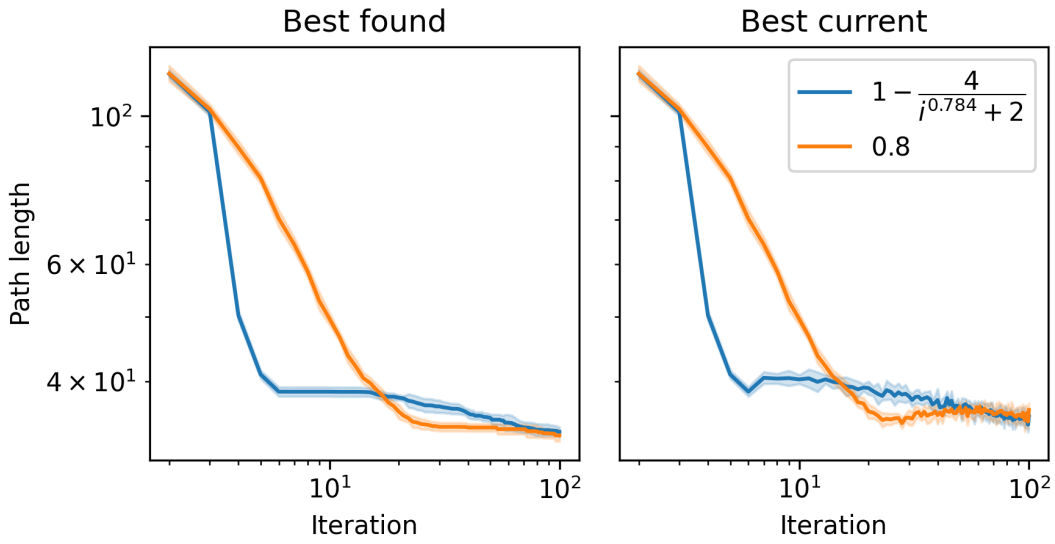

Figure 5.4: Comparison of constant and dynamic ρ values, 10 ACO instances for each value. The problem instance is 1000 random cities, other parameter values are $\alpha = 1$, $\beta = 2$, $m = 20$, $Q = 1$ and $\sigma = 1$.

5.5 Reinforcement Learning and ACO

Reinforcement Learning (RL) is a subfield of machine learning that focuses on training agents to make decisions by interacting with an environment. In RL, an agent learns to choose the best actions based on the feedback it receives in the form of rewards or penalties. The primary goal of the agent is to maximize the cumulative reward over time [59].

One of the fundamental algorithms in RL is Q-learning, an off-policy, model-free algorithm that estimates the expected value of taking an action in a specific state. Q-learning enables the agent to learn an optimal policy for decision-making by updating the estimated action values iteratively, based on the rewards received and the discounted value of future states. The algorithm converges to the optimal policy when given enough time [63].

Ant-Q is a variant of the ACO algorithm that incorporates concepts from Q-learning, forming a hybrid between reinforcement learning and ACO. In Ant-Q, artificial ants act as agents, learning to choose the best actions by updating a pheromone matrix based on the quality of the solutions they find [32, 23].

Reinforcement learning has gained significant attention in recent years, thanks to its success in various domains, such as robotics, game playing, and control systems [59]. However, with a better understanding of ACO and its parameters, the algorithm also has the potential to deliver comparable results. Future research efforts could focus on bringing recent insights from RL to ACO.

5.6 The Real World

In this section, we discuss the results of implementing the ACO algorithm for urban waste management in The City of Edinburgh, focusing on two instances: communal clear glass bins and communal food waste bins. Our objective is to understand the performance of the ACO algorithm in real-world applications and compare its solutions to those provided by the current software used by The City of Edinburgh, RouteSmart [47].

As detailed in Subsection 3.4, we experimented with two different heuristics for constructing the problem: Euclidean distances and non-planar real distances. Figures 4.10 and 4.11 show the tours generated by the ACO algorithm using these heuristics for the communal clear glass bins in Edinburgh. The non-planar heuristic yields a slightly better tour length of 71.1 km compared to 73.4 km with the Euclidean heuristic. This improvement suggests that incorporating real-world distances into the problem statement can result in more accurate and efficient routes.

In the second experiment, we focus on the City Centre of Edinburgh, a highly non-planar and complex area characterized by tunnels, one-directional streets, bridges, and dead-end roads. In this case, using a non-planar heuristic significantly improves the results from 37.9 km to 27.2 km, as shown in Figures 4.12 and 4.13. This substantial improvement underscores the importance of using accurate heuristics for problem instances with a high degree of non-planarity.

When comparing the ACO algorithm's solutions to those provided by RouteSmart, the current software used by The City of Edinburgh, it is essential to consider several factors. RouteSmart is a commercial software package that has been specifically designed and optimized for waste collection routing [2]. It incorporates advanced algorithms and heuristic approaches, as demonstrated in [53] The last public information on RouteSmart's functionality dates back to 2008, and it is expected that the software may not have changed significantly since then.

RouteSmart is proprietary software, thus its underlying algorithms and methods are not openly accessible. Furthermore, RouteSmart performs multi-objective optimization that caters to various objectives in waste collection routing (e.g. such as avoiding certain times). It is important to note that the ACO algorithm can be extended to handle multi-objective problems through scalarization techniques, offering a more comprehensive optimization solution. Additionally, the software package may include numerous features that may not be necessary for a city the size of Edinburgh, potentially leading to increased complexity and cost.

The ACO algorithm developed in this dissertation is open-source, allowing for transparency and the possibility of community-driven improvements. While the ACO algorithm has demonstrated its ability to generate efficient routes in our experiments, it may not yet be on par with the performance of a dedicated routing software like RouteSmart. However, the results obtained in the experiments suggest that there is potential for improvement and further development of the ACO algorithm to tackle real-world urban waste management problems effectively.

The impact of this work lies in demonstrating the applicability of the ACO algorithm to real-world problems, particularly in the context of urban waste management. By exploring different heuristics and problem instances, we have gained valuable insights into the performance and adaptability of the ACO algorithm. These insights can be used to guide future research and development efforts aimed at improving the efficiency and effectiveness of ACO and other metaheuristics for solving complex, non-planar optimization problems.

Future work in this area may include exploring additional heuristics, refining the ACO algorithm for improved performance, and extending the algorithm to handle multi-objective optimization through scalarization techniques. By considering multiple objectives in the optimization process, the ACO algorithm can better cater to the diverse needs and constraints of urban waste management systems. Additionally, further research can be conducted on integrating ACO with other optimization algorithms or techniques, such as local search or hybrid algorithms, to achieve even better results.

Chapter 6

Conclusion and Further Study

6.1 Conclusion

In this dissertation, we investigated the ACO algorithm and its application to TSP and real-world urban waste management problems. Through a series of experiments, we gained valuable insights into the impact of various parameter values, heuristics, and problem instances on the performance of the ACO algorithm.

The experiments on the TSP revealed the importance of understanding the role of parameters such as α, β, ρ, and σ in the ACO algorithm. By carefully tuning these parameters, we were able to optimize the algorithm's performance and improve its ability to solve the TSP. We also examined the effect of varying the number of iterations on the performance of the algorithm and discovered an interesting relationship between the optimal evaporation rate and the number of iterations.

The real-world application to urban waste management has demonstrated the potential of the ACO algorithm in generating efficient waste collection routes for The City of Edinburgh. We found that using non-planar heuristics significantly improved the algorithm's performance, particularly in complex, non-planar areas such as the city centre. Although this ACO implementation may not yet rival the performance of dedicated routing software like RouteSmart, the open-source nature and adaptability of the algorithm provide a promising foundation for further development and improvement.

This work contributes to the understanding of the applicability and potential of metaheuristics, particularly ACO, for solving real-world optimization problems in the context of urban waste management. Our findings can be used to guide future research and development efforts aimed at enhancing the efficiency and effectiveness of ACO and other metaheuristics for solving complex, non-planar optimization problems.

6.2 Further Study

There are several avenues for further research in this area. One direction is to explore additional heuristics and refine the ACO algorithm for improved performance in solving the TSP and other optimization problems. This includes investigating the impact of dynamic parameter values, such as the evaporation rate, on the algorithm's performance and adaptability. Additionally, the development of more sophisticated heuristics for real-world problems, such as urban waste management, may lead to even better results.

Another area of interest is the extension of the ACO algorithm to handle multi-objective optimization problems through scalarization techniques. By considering multiple objectives in the optimization process, the ACO algorithm can better cater to the diverse needs and constraints of urban waste management systems and other real-world applications.

Further research can also focus on integrating ACO with other optimization algorithms or techniques, such as local search or hybrid algorithms, to achieve even better results. With a better understanding of the effect of the parameters of ACO, the combination of different optimization approaches may lead to the development of more powerful and versatile metaheuristics, capable of solving a wide range of complex optimization problems.

Lastly, the application of ACO to other real-world problems beyond urban waste management, such as other vehicle routing, logistics, and supply chain optimization, offers the opportunity to explore the versatility and potential of the algorithm in various domains. By demonstrating the effectiveness of ACO in different settings, we can contribute to the broader understanding and application of metaheuristics for real-world optimization challenges.

Bibliography

[1] Interface runnable. https://docs.oracle.com/javase/7/docs/api/java/lang/Runnable.html. Accessed: 2023-04-07.

[2] Routesmart - intelligent routing software. http://web.archive.org/web/20230406081257/https://www.routesmart.com/. Accessed: 2023-04-07.

[3] Daniel Angus and Clinton Woodward. Multiple objective ant colony optimisation. *Swarm intelligence*, 3:69–85, 2009.

[4] David L Applegate, Robert E Bixby, Vašek Chvátal, and William J Cook. The traveling salesman problem. In *The Traveling Salesman Problem*. Princeton university press, 2011.

[5] Erfan Babaee Tirkolaee, Iraj Mahdavi, Mir Mehdi Seyyed Esfahani, and Gerhard-Wilhelm Weber. A hybrid augmented ant colony optimization for the multi-trip capacitated arc routing problem under fuzzy demands for urban solid waste management. *Waste management & research*, 38(2):156–172, 2020.

[6] Brian W Baetz. Optimization/simulation modeling for waste management capacity planning. *Journal of Urban Planning and Development*, 116(2):59–79, 1990.

[7] Per Bak. *How nature works: the science of self-organized criticality*. Springer Science & Business Media, 2013.

[8] Jillian Beardwood, John H Halton, and John Michael Hammersley. The shortest path through many points. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 55, pages 299–327. Cambridge University Press, 1959.

[9] John E Bell and Patrick R McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48, 2004.

[10] George Bilchev and Ian C Parmee. The ant colony metaphor for searching continuous design spaces. In *Evolutionary Computing: AISB Workshop Sheffield, UK, April 3–4, 1995 Selected Papers*, pages 25–39. Springer Berlin Heidelberg, 1995.

[11] Christian Blum. Beam-aco—hybridizing ant colony optimization with beam search: An application to open shop scheduling. *Computers & Operations Research*, 32(6):1565–1591, 2005.

[12] Bernd Bullnheimer, Richard Hartl, and Christine Strauss. A new rank based version of the ant system - a computational study. *Central European Journal of Operations Research*, 7:25–38, 01 1999.

[13] Joao L Caldeira, Ricardo C Azevedo, Carlos A Silva, and Joao MC Sousa. Supply-chain management using aco and beam-aco algorithms. In *2007 IEEE International Fuzzy Systems Conference*, pages 1–6. IEEE, 2007.

[14] Vašek Chvátal, William Cook, George B Dantzig, Delbert R Fulkerson, and Selmer M Johnson. Solution of a large-scale traveling-salesman problem. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 7–28, 2010.

[15] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[16] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

[17] J L Deneubourg, Serge Aron, Simon Goss, and Jacques M Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of insect behavior*, 3:159–168, 1990.

[18] Wu Deng, Junjie Xu, and Huimin Zhao. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE access*, 7:20281–20292, 2019.

[19] Gianni Di Caro and Marco Dorigo. *Ant colony optimization and its application to adaptive routing in telecommunication networks*. PhD thesis, PhD thesis, Faculté des Sciences Appliquées, Université Libre de Bruxelles . . . , 2004.

[20] James E Doran and Donald Michie. Experiments with the graph traverser program. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 294(1437):235–259, 1966.

[21] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[22] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical computer science*, 344(2-3):243–278, 2005.

[23] Marco Dorigo and Luca Maria Gambardella. A study of some properties of ant-q. In *Parallel Problem Solving from Nature—PPSN IV: International Conference on Evolutionary Computation—The 4th International Conference on Parallel Problem Solving from Nature Berlin, Germany, September 22–26, 1996 Proceedings 4*, pages 656–665. Springer, 1996.

[24] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[25] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[26] Marco Dorigo and Thomas Stützle. *Ant colony optimization: overview and recent advances*. Springer, 2019.

[27] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[28] Adam Erskine and J Michael Herrmann. Crips: Critical particle swarm optimisation. In *Artificial Life Conference Proceedings*, pages 207–214. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2015.

[29] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Multi-objective, decentralized dynamic virtual machine consolidation using aco metaheuristic in computing clouds. *arXiv preprint arXiv:1706.06646*, 2017.

[30] C-N Fiechter. A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics*, 51(3):243–267, 1994.

[31] C Fountas and A Vlachos. Ant colonies optimization (aco) for the solution of the vehicle routing problem (vrp). *Journal of information and optimization sciences*, 26(1):135–142, 2005.

[32] Luca M Gambardella and Marco Dorigo. Ant-q: A reinforcement learning approach to the traveling salesman problem. In *Machine learning proceedings 1995*, pages 252–260. Elsevier, 1995.

[33] Simon Goss, Serge Aron, Jean-Louis Deneubourg, and Jacques Marie Pasteels. Self-organized shortcuts in the argentine ant. *Naturwissenschaften*, 76(12):579–581, 1989.

[34] P P Grassé and Ch Noirot. L'évolution de la symbiose chez les isoptères. *Experientia*, 15:365–372, 1959.

[35] Şaban Gülcü, Mostafa Mahi, Ömer Kaan Baykan, and Halife Kodaz. A parallel cooperative hybrid method based on ant colony optimization and 3-opt algorithm for solving traveling salesman problem. *Soft Computing*, 22:1669–1685, 2018.

[36] Michael Guntsch and Martin Middendorf. Solving multi-criteria optimization problems with population-based aco. In *Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings 2*, pages 464–478. Springer, 2003.

[37] Walter J Gutjahr and Marion S Rauner. An aco algorithm for a dynamic regional nurse-scheduling problem in austria. *Computers & Operations Research*, 34(3):642–666, 2007.

[38] Michael Held and Richard M Karp. The traveling-salesman problem and minimum spanning trees. *Operations Research*, 18(6):1138–1162, 1970.

[39] J Michael Herrmann, Adam Price, and Thomas Joyce. 3. ant colony optimization and reinforcement learning. In *Computational Intelligence*, pages 45–62. De Gruyter, 2020.

[40] Pierre C Hohenberg and Bertrand I Halperin. Theory of dynamic critical phenomena. *Reviews of Modern Physics*, 49(3):435, 1977.

[41] Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, 52:2191–2233, 2019.

[42] Nikolaos V Karadimas, Katerina Papatzelou, and Vassili G Loumos. Optimal solid waste collection routes identified by the ant colony system algorithm. *Waste management & research*, 25(2):139–147, 2007.

[43] R Kavitha, D Kiruba Jothi, K Saravanan, Mahendra Pratap Swain, José Luis Arias Gonzáles, Rakhi Joshi Bhardwaj, Elijah Adomako, et al. Ant colony optimization-enabled cnn deep learning technique for accurate detection of cervical cancer. *BioMed Research International*, 2023, 2023.

[44] Morten Lovbjerg and Thiemo Krink. Extending particle swarm optimisers with self-organized criticality. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1588–1593. IEEE, 2002.

[45] Oscar Montiel-Ross, Nataly Medina-Rodriguez, Roberto Sepulveda, and Patricia Melin. Methodology to optimize manufacturing time for a cnc using a high performance implementation of aco. *International Journal of Advanced Robotic Systems*, 9(4):121, 2012.

[46] Jonathan Moss and Colin G Johnson. An ant colony algorithm for multiple sequence alignment in bioinformatics. In *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Roanne, France, 2003*, pages 182–186. Springer, 2003.

[47] Angus Murdoch. Request for edinburgh waste management information, Apr 2023. Email communications.

[48] Dheeraj Pal, Pratima Verma, Divya Gautam, and Priyanka Indait. Improved optimization technique using hybrid aco-pso. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, pages 277–282. IEEE, 2016.

[49] Riccardo Poli. Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008:1–10, 2008.

[50] Sarifah Putri Raflesia and Anugrah K. Pamosoaji. A novel ant colony optimization algorithm for waste collection problem. In *2019 4th International Conference on Information Technology, Information Systems and Electrical Engineering (ICITISEE)*, pages 413–416, 2019.

[51] Teng Ren, Tianyu Luo, Binbin Jia, Bihao Yang, Ling Wang, and Lining Xing. Improved ant colony optimization for the vehicle routing problem with split pickup and split delivery. *Swarm and Evolutionary Computation*, 77:101228, 2023.

[52] Surya Sahoo, Seongbae Kim, Byung-In Kim, Bob Kraas, and Alexander Popov Jr. Routing optimization for waste management. *Interfaces*, 35(1):24–36, 2005.

[53] Robert Shuttleworth, Bruce L Golden, Susan Smith, and Edward Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network, 2008.

[54] Eric Sigel, Bruce Denby, and Sylvie Le Hégarat-Mascle. Application of ant colony optimization to adaptive routing in a leo telecommunications satellite network. In *Annales des télécommunications*, volume 57, pages 520–539. PRESSES POLYTECHNIQUES ROMANDES, 2002.

[55] Kenneth Sörensen. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1):3–18, 2015.

[56] Thomas Stützle and Marco Dorigo. Aco algorithms for the quadratic assignment problem. *New ideas in optimization*, 33, 1999.

[57] Thomas Stutzle and Marco Dorigo. A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactions on evolutionary computation*, 6(4):358–365, 2002.

[58] Thomas Stützle and Holger H Hoos. Max–min ant system. *Future generation computer systems*, 16(8):889–914, 2000.

[59] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[60] E-G Talbi, Olivier Roux, Cyril Fonlupt, and Denis Robillard. Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4):441–449, 2001.

[61] Ahamed Fayeez Tuani, Edward Keedwell, and Matthew Collett. Heterogenous adaptive ant colony optimization with 3-opt local search for the travelling salesman problem. *Applied Soft Computing*, 97:106720, 2020.

[62] Christine L Valenzuela and Antonia J Jones. Estimating the held-karp lower bound for the geometric tsp. *European journal of operational research*, 102(1):157–175, 1997.

[63] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[64] Fanlei Yan. Autonomous vehicle routing problem solution based on artificial potential field with parallel ant colony optimization (aco) algorithm. *Pattern recognition letters*, 116:195–199, 2018.

[65] Yuru Zhang, Chenyang Wang, Hui Li, Xiaodong Su, Ming Zhao, and Nan Zhang. An improved 2-opt and aco hybrid algorithm for tsp. In *2018 Eighth International*

Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), pages 547–552, 2018.

[66] Xiangbing Zhou, Hongjiang Ma, Jianggang Gu, Huiling Chen, and Wu Deng. Parameter adaptation-based ant colony optimization with dynamic hybrid mechanism. *Engineering Applications of Artificial Intelligence*, 114:105139, 2022.

Appendix A

Python and Java Code

A.1 Finding the Best ρ for Each Iteration

In the given code snippet, two Python functions are used to analyze a DataFrame df in order to find the best rho values that minimize the `bestLength` for each iteration. The code is as follows:

```
def average(df, pos, step):
    tmp = df.loc[pos:pos+step]
    return (pos, tmp['bestLength'].mean())

def best_rho_for_all_iters(step, df):
    best_rhos = []
    for i in range(2, df.iteration.max()+1):
        df_i = df[df['iteration'] == i]
        df_i = df_i[['rho', 'bestLength']].set_index('rho')
        l, pos = [], 0
        while pos + step <= 1:
            pos += 0.01
            l.append(average(df_i, pos, step))
        best_rhos.append(min(l, key=lambda t: t[1])[0])
    return best_rhos
```

The first function, `average(df, pos, step)`, takes a `DataFrame`, a position, and a step size as input. It calculates the mean of the `bestLength` values within a specified range in the DataFrame. The range is determined by the position pos and the step size step.

The second function, `best_rho_for_all_iters(step, df)`, processes the DataFrame `df` to compute optimal `rho` values that minimize `bestLength` for all iterations and returns a list containing the best `rho` values.

A.2 Creating TSP from the Communal Bins of Edinburgh

This is the structure of the Python script that creates and saves two types of adjacency matrices for a set of geographical locations, where each location represents a bin. The script takes optional command-line arguments for bin type and ward name to filter bins. It uses the OSMnx library to handle OpenStreetMap data and NetworkX for graph manipulation.

`create_graph()`:

```
def create_graph():
    return ox.load_graphml("data/edinburgh_graph.graphml")
```

Loads a pre-built graph from a GraphML file and returns it.

`heu(node, goal)`:

```
def heu(node, goal):
    return ox.distance.euclidean_dist_vec(G.nodes[node]['x'],
        G.nodes[node]['y'], G.nodes[goal]['x'], G.nodes[goal]['y'])
```

Calculates the Euclidean distance between two nodes in the graph as a heuristic function for the A* algorithm.

`create_map_nonplanar(node_ids)`:

```
def create_map_nonplanar(node_ids):
    ...
    adjacency_matrix = np.zeros((len(node_ids), len(node_ids)))
    ...
    adjacency_matrix[i, j] = nx.algorithms.shortest_paths
        .astar.astar_path_length(G, node_ids[i], node_ids[j],
        heuristic=heu , weight='length')
    ...
    return (normalize(adjacency_matrix, axis=1, norm='l1'), node_ids)
```

Builds a non-planar adjacency matrix using the A* algorithm to find the shortest path lengths between nodes in the graph. Normalizes the adjacency matrix.

`create_map(node_ids)`:

```
def create_map(node_ids):
    adjacency_matrix = np.zeros((len(node_ids), len(node_ids)))
    ...
    adjacency_matrix[i, j] = ox.distance.euclidean_dist_vec(
        G.nodes[node_ids[i]]['x'], G.nodes[node_ids[i]]['y'],
        G.nodes[node_ids[j]]['x'], G.nodes[node_ids[j]]['y'])
    ...
    return (normalize(adjacency_matrix, axis=1, norm='l1'))
```

Constructs a planar adjacency matrix using Euclidean distances between nodes, and normalizes the matrix.

`main()`:

```
def main():
    ...
    global G
    G = create_graph()
    ...
    node_ids = list(set(ox.distance.nearest_nodes(G, lon, lat,
        return_dist=False)))
    adjacency_matrix_nonplanar, node_ids =
        create_map_nonplanar(node_ids)
    adjacency_matrix_euclidean = create_map(node_ids)
    ...
    np.savetxt(save_file_euclidean, adjacency_matrix_euclidean,
        fmt='%1.8f', delimiter=",")
    np.savetxt(save_file_nonplanar, adjacency_matrix_nonplanar,
        fmt='%1.8f', delimiter=",")
    ...
```

Parses command-line arguments for bin type and ward name, loads the graph, filters bins based on input, obtains node IDs for the bins, calls `create_map_nonplanar()` and `create_map`() to generate adjacency matrices, and saves the matrices along with node IDs to CSV files.

The script first filters bins based on the bin type and ward name (if provided) and then computes the nearest nodes for these bins. Next, it generates a non-planar adjacency matrix using the A* algorithm for shortest paths and a planar adjacency matrix using Euclidean distances. The matrices are normalized and saved to separate CSV files, along with the node IDs.

A.3 ACO in Java

A.3.1 Move Probability from City to City

The `MoveProb` function calculates the probability of an ant moving from the current city `cityX` to all other cities in the problem domain, given a boolean array visited that indicates whether a city has been visited.

```
private double[] MoveProb(int cityX, boolean[] visited)
{
    double[] attractiveness = new double[numCities];
    double sum = 0.0;
    for (int i = 0; i <= attractiveness.length - 1; i++)
    {
        if (i == cityX)
        {
            attractiveness[i] = 0.0;
        }
```

```
        else if (visited[i])
        {
            attractiveness[i] = 0.0;
        }
        else
        {
            attractiveness[i] = Math.pow(pheromones[cityX][i],
                alpha) * Math.pow((1.0 / Distance(cityX, i)), beta);
            if (attractiveness[i] < 0.0001)
            {
                attractiveness[i] = 0.0001;
            }
            else if (attractiveness[i] > (Double.MAX_VALUE
                / (numCities * 100)))
            {
                attractiveness[i] = Double.MAX_VALUE
                    / (numCities * 100);
            }
        }
        sum += attractiveness[i];
    }

    double[] prob = new double[numCities];
    for (int i = 0; i <= prob.length - 1; i++)
    {
        prob[i] = attractiveness[i] / sum;
    }
    return prob;
}
```

The function initializes an array `attractiveness` of length `numCities` and iterates over all cities. If the city is the current city or has been visited, its attractiveness is set to 0. For unvisited cities, attractiveness is calculated based on pheromone levels and the inverse of the distance between cities, raised to the powers of `alpha` and `beta`, respectively. The calculated attractiveness is then clamped between a minimum value of 0.0001 and a maximum value derived from `Double.MAX_VALUE` and the number of cities. The sum of all attractiveness values is calculated in the process.

Afterwards, a `prob` array is created to store the probability values for each city. These probabilities are obtained by dividing the attractiveness value of each city by the sum of attractiveness values. Finally, the `prob` array is returned, representing the probability distribution for an ant to move from the current city to all other cities.

A.3.2 Pheromone Update Rule

The `UpdatePheromones` function is responsible for modifying the pheromone matrix after each iteration to guide the ants towards more promising solutions.

```
private void UpdatePheromones()
{
    for (int i = 0; i <= pheromones.length - 1; i++)
    {
        for (int j = i + 1; j <= pheromones[i].length - 1; j++)
            // change if graph is directed
        {
            double sum = 0.0;

            for (int k = 0; k <= ants.length - 1; k++) {
                double length = Length(ants[k]);
                if (EdgeInTrail(i, j, ants[k])) {
                    double add = Q * (1.0 / length);
                    if (length <= bestLength) {
                        add *= sigma;
                    }
                    sum += add;
                }
            }
            pheromones[i][j] = rho * pheromones[i][j] + (1 - rho) * sum;

            if (pheromones[i][j] < 0.00001)
                pheromones[i][j] = 0.00001;
            else if (pheromones[i][j] > 1000000.0)
                pheromones[i][j] = 1000000.0;

            pheromones[j][i] = pheromones[i][j];
                // change if graph is directed
        }
    }
}
```

The function iterates over all pairs of cities and calculates the total pheromone deposit by all ants for the given pair of cities. If the edge between the cities is part of an ant's trail, the pheromone deposit is calculated based on the inverse of the trail's length, scaled by a constant factor `Q`. If the trail's length is smaller than or equal to the best length found so far, the pheromone deposit is further multiplied by a scaling factor `sigma`.

The pheromone matrix is then updated using a weighted sum of the existing pheromone value and the calculated pheromone deposit. The weight is determined by the evaporation factor `rho`. The updated pheromone value is clamped between a minimum of 0.00001 and a maximum of 1000000.0. The pheromone values for the reverse direction are set to be equal to the forward direction, assuming an undirected graph. This can be changed for directed graphs.

Appendix B

Additional Figures

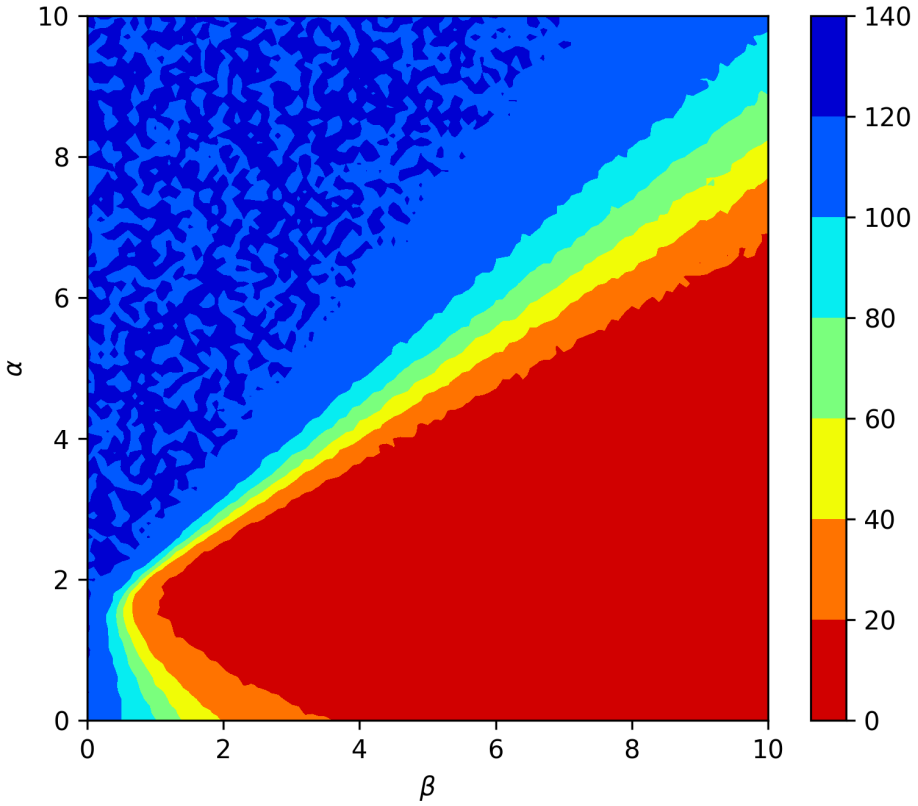

Figure B.1: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (0, 10)$, after 20 iterations on 250 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

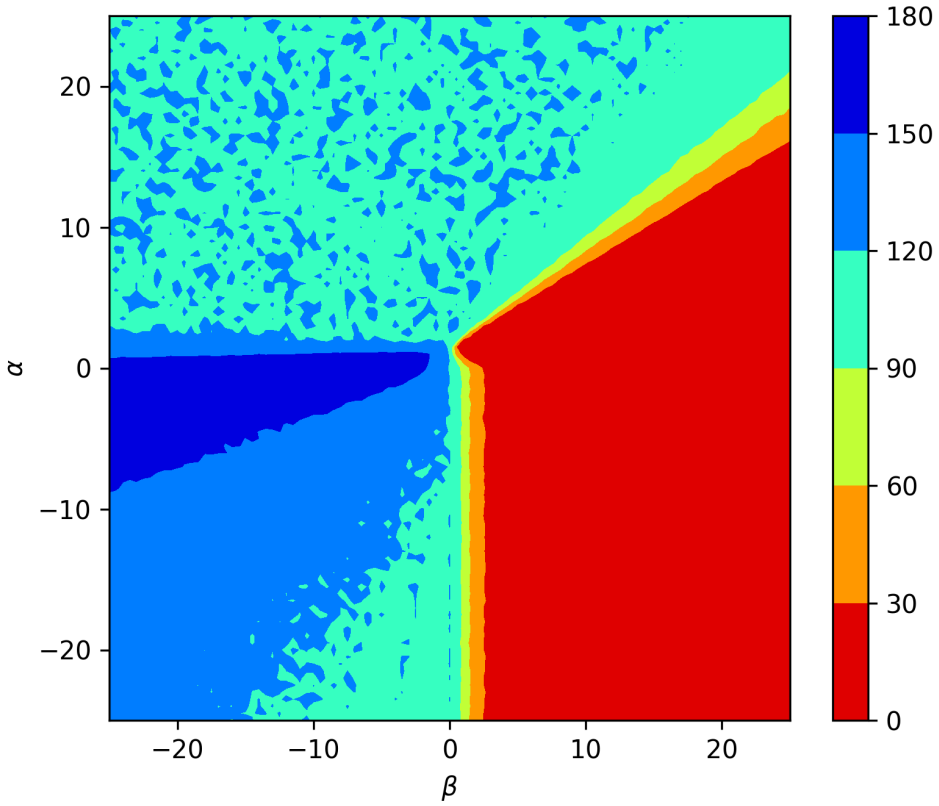

Figure B.2: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (-25, 25)$, after 10 iterations on 250 random cities. Other parameter values are $\rho = 0.8$, $m = 20$, $Q = 1$ and $\sigma = 1$.

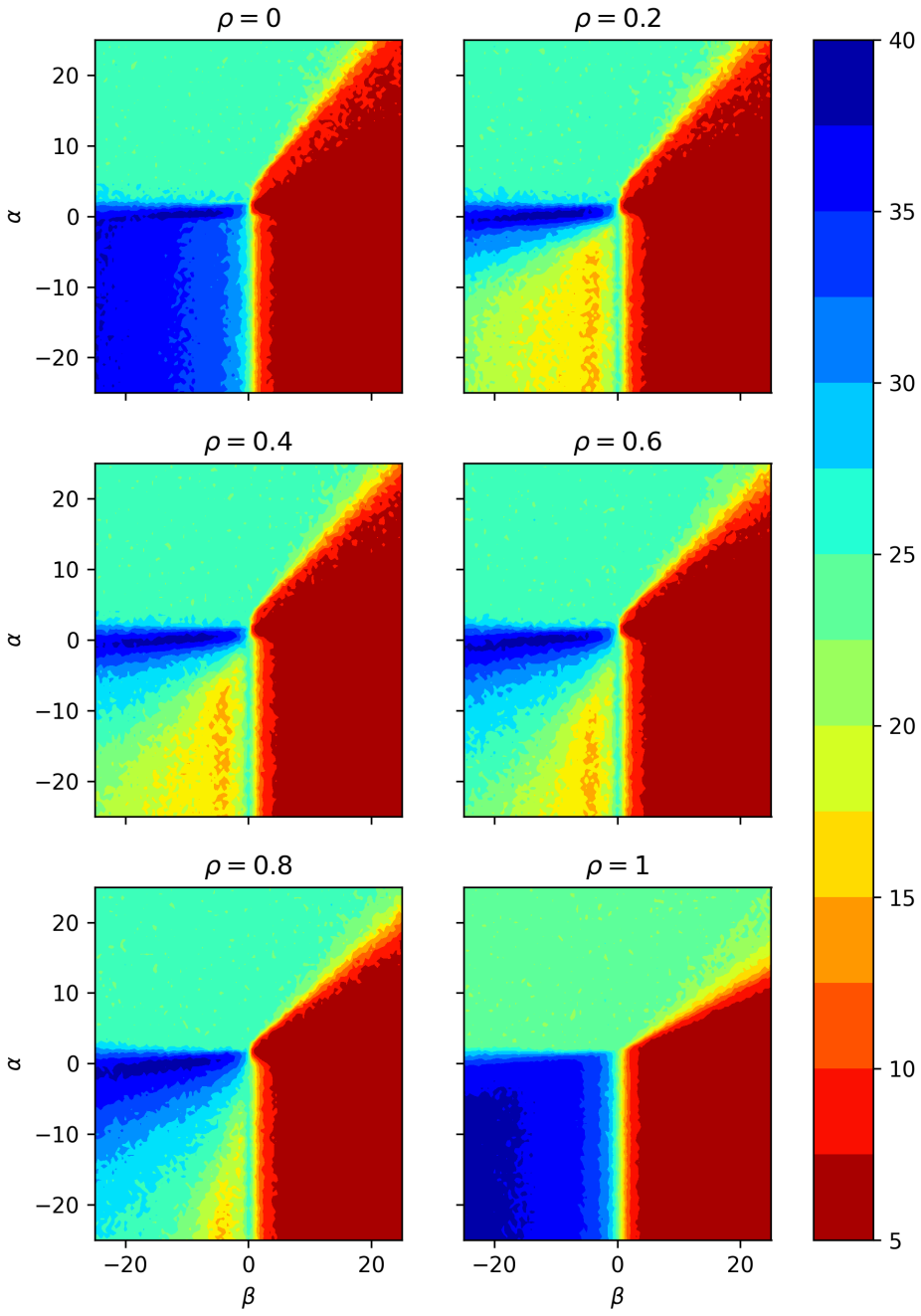

Figure B.3: Filled contour plot of the best length of each instance of ACO for each value of $\alpha, \beta \in (-25, 25)$, for different ρ after 10 iterations on 250 random cities. Other parameter values are $m = 20$, $Q = 1$ and $\sigma = 1$.