NTNU – Trondheim
Norwegian University of
Science and Technology

A Combined Swarm System for the Urban Transit Routing Problem

Hanne Gunby
Susanne Gustavsen

Master of Science in Informatics
Submission date: May 2015
Supervisor: Agnar Aamodt, IDI
Co-supervisor: Anders Kofod-Petersen, IDI

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

Abstract

The goal of this thesis is to develop a system that improves urban transit networks. A good transit network can reduce the number of vehicles on the road as people will favor public transport over private transportation. This will eventually help reduce congestion and environmental emissions.

The Urban Transit Routing Problem (UTRP) concerns the creation of route networks. UTRP is a complex and multiconstrained problem, in which the creation of route networks can both be challenging and time consuming. Metaheuristics like swarm intelligence methods have proven to be effective of finding sufficient solutions to these types of NP-hard problems. In this contribution, a swarm inspired optimization system is designed and presented, aiming to create efficient solutions to the UTRP. The proposed system uses an ant colony approach with, unlike previous techniques, additional attributes inspired by bee colony optimization and particle swarm optimization.

A structured literature review is conducted to synthesize the relevant primary studies. All retrieved results are presented and analyzed. Further, because metaheuristics require good parameter values to solve concrete problems optimally, a thorough review and justification of each selected parameter is documented. This documentation will contribute in providing a starting point for potential future research. A comparison of a standard ant colony optimization (ACO) implementation is performed to determine whether the proposed system improves the standard ACO. To demonstrate the performance of the proposed system, obtained results are compared against results published in the literature. Results are compared on the basis of Mandl's benchmark problem, which is a widely investigated and acknowledged benchmark problem. The proposed system is also tested on larger networks, more similar to real transit networks, to validate whether the proposed system supports larger networks as input. This thesis will also report how the usage of the graph database Neo4j has affected the development and quality of the proposed system.

Comparison of obtained results with the standard ACO implementation and other published results are promising, especially regarding the average traveling time per transit user.

Sammendrag

Målet med denne oppgaven er å utvikle et system for å optimalisere rutenettverk i byer, for å videre gjøre offentlig transport mer praktisk for passasjerer. Et godt rutenettverk kan redusere antall biler på veien ved at passasjerer velger kollektivtransport fremfor egne transportmidler. Dette vil igjen bidra med å redusere trafikkøer og miljøutslipp.

"Urban Transit Routing Problems" (UTRP) omhandler konstruksjon av rutenettverk for kollektivtransport. UTRP er et komplekst og "multiconstrained" problem, der konstruksjonen av rutenettene kan være både kompliserte og tidkrevende. Metaheuristiske metoder, som sverm intelligens, har vist seg å være effektive for å finne tilstrekkelige løsninger på denne typen "NP-hard" problemer. I dette bidraget, er et sverm inspirert optimaliseringssystem utformet og presentert, med formålet om å skape effektive løsninger til UTRP. Det foreslåtte systemet bruker en "ant colony" tilnærming med, i motsetning til tidligere løsninger, tilleggsattributter inspirert av "bee colony optimalization" og "particle swarm optimalization".

Et strukturert litteratursøk er gjennomført for å syntetisere relevante primærstudier. Alle resultater blir presentert og analysert. Videre, fordi metaheuristics krever gode parameterverdier for å løse konkrete problemer optimalt, er en grundig gjennomgang og begrunnelse for hvert valgte parameter dokumentert. Denne dokumentasjonen kan være et utgangspunkt for potensiell fremtidig forskning. En sammenligning med en standard "ant colony optimalization" (ACO) implementasjon er gjennomført for å vise om det foreslåtte systemet forbedrer en standard ACO. For å undersøke ytelsen til det foreslåtte systemet, sammenlignes de oppnådde resultatene mot resultater publisert i litteraturen. Resultatene er sammenlignet på basis av "Mandl's benchmark problem", som er et anerkjent og mye brukt "benchmark" problem. Det foreslåtte systemet er også testet på større nettverk, mer likt nettverk i ekte byer, for å validere om det foreslåtte systemet støtter større nettverk som input. Denne oppgaven vil også rapportere hvordan bruken av grafdatabasen Neo4j har påvirket utviklingen og ytelsen av det foreslåtte systemet.

Sammenligningen av oppnådde resultater med den standard ACO implementasjonen og andre publiserte resultater er lovende, spesielt når det gjelder gjennomsnittlig reisetid per reisende.

Preface

This document and its associated source-code is the end product of the Masters Thesis of Hanne Gunby and Susanne Gustavsen at the Norwegian University of Science and Technology (NTNU).

Acknowledgments

Thanks to Anders Kofod-Petersen and Jo Skjermo for valuable feedback and advice throughout the process.

Thanks to Agnar Aamodt for always keeping his door open and answering questions.

Thanks to Torkil Rein Gustavsen for introducing us to Cloud Computing.

Thanks to friends and fellow students for helpful discussions.

Finally, thanks to our family for endless love and support.

List of Tables

4.1 The parameters to be set for the system 44

5.1 Parameters to be tested, and their default value to be held constant while the other parameters are tested 54

5.2 Results from the parameter settings experiment 55

5.3 Running time in seconds for different candidate values of parameters s and i 56

5.4 The best route set, having four routes, constructed by the generic ACO implementation and the proposed system. 59

5.5 Comparing the best route set, having four routes, produced by the proposed system, with route sets constructed by other approaches. . 60

5.6 Comparing the best route set on Mandl's Network, having six routes, produced by the proposed system with route sets constructed by other approaches. 62

5.7 Comparing the best route set on Mandl's Network, having seven routes, produced by the proposed system with route sets constructed by other approaches. 64

5.8 Comparing the best route set on Mandl's Network, having eight routes, produced by the proposed system with route sets constructed by other approaches. 66

5.9 Networks with properties from the supplementary material from Mumford [2013]. 67

5.10 Comparing the run time in seconds for Method 1 and Method 2 on the Mumford0 network. 67

5.11 Results produced for the instances added as supplementary material of Mumford [2013] 68

6.1 A selection of the average $TOTFIT$ with different parameter values for CA and AF 71

6.2 Average produced values of the performance criteria for route set sizes four, five, six and seven on Mandl's Network 73

6.3 Number of RelationshipTypes that will be generated using different route set sizes, with 125 iterations and a swarm size of 50. 75

6.4 Average Total Fitness for all tests cases. 1-4 are the results of 50 runs, 5-6 are the results of 10 runs. 76

6.5 Average run time in seconds of 10 runs using the Mumford Networks 76

6.6 Average run time in seconds of 50 runs for each route set size on the Mandl Network 77

A.1 Quality criteria scoring 93

A.2 Final selected literature 94

B.1 Coordinates for Mandl's Network 104

B.2 Travel times for Mandl's Network 104

B.3 Demand for Mandl's Network 105

C.1 Steps with the corresponding results from the parameter settings experiment (parameter s, i and E) 108

C.2 Steps with the corresponding results from the parameter settings experiments (parameter CA, AF and p_b) 108

C.3 Comparing the route sets of ACO and CSS, having four routes (all results) 109

List of Figures

2.1 An illustration of how ants adapt to change 22

2.2 Illustration of particles in a 2D-space in an early iteration of the PSO algorithm (exploring) 25

2.3 Illustration of particles in a 2D-space in a late iteration of the PSO algorithm (exploiting) 25

2.4 Eulers original drawing of the Seven Bridges of Königsberg 28

3.1 Illustration of Mandl's Network as a graph 39

5.1 Illustration of the best route set on Mandl's Network, having four routes, constructed by the proposed system 59

5.2 Illustration of the best route set on Mandl's Network, having six routes, constructed by the proposed system 61

5.3 Illustration of the best route set on Mandl's Network, having seven routes, constructed by the proposed system 63

5.4 Illustration of the best route set on Mandl's Network, having eight routes, constructed by the proposed system 65

6.1 Average TOTFIT value of 10 runs at each iteration for ACO and CSS 70

6.2 A fragment of the best route set, having four route sets, constructed by the proposed algorithm including transfer times in minutes between each node. 73

Contents

1 Introduction **15**

1.1 Background and motivation 15

1.2 Goal and research questions 16

1.3 Research methods 17

1.4 Thesis overview 18

2 Theory and Background **21**

2.1 Swarm intelligence 21

2.1.1 Ant colony optimization 21

2.1.2 Bee colony optimization 23

2.1.3 Particle swarm optimization 24

2.2 Vehicle Routing Problem 26

2.2.1 Urban Transit Network Design Problem 26

2.3 Graph databases 27

2.3.1 Neo4j 28

3 Preparatory Research **31**

3.1 Refining the research topic 31

3.2 Structured literature review 31

3.3 Related work 32
3.3.1 Vehicle Routing Problems 32
3.3.2 Urban Transit Network Design Problems 34
3.3.3 Discussion 36
3.3.4 Conclusion 37
3.4 Problem statement 38

4 The Proposed System **41**

4.1 Combined Swarm System 41
4.2 Development environment 42
4.3 System assessments 42
4.3.1 Constraints 42
4.3.2 Evaluation criteria 43
4.4 Initialization 44
4.4.1 Parameters 44
4.4.2 Input data 44
4.4.3 Network generation 45
4.5 Generating Combined Swarm colony 45
4.6 Selecting start node 46
4.7 Selecting next nodes 46
4.8 Creating the route set 48
4.9 Evaluation 48
4.9.1 Removing ants that did not fulfill the constraints 48
4.9.2 Calculating Total Fitness 49
4.10 Selecting ants to be followed 51
4.11 Pheromone evaporation 51

5 Experiments and Results **53**

5.1 Parameter settings 53

5.1.1 Experimental plan 53

5.1.2 Experimental setup 53

5.1.3 Experimental results 55

5.1.4 Selecting final parameters 55

5.2 Performance comparison 57

5.2.1 Experimental plan 57

5.2.2 Experimental setup 57

5.2.3 Experimental results 58

5.3 Network expansion 66

5.3.1 Experimental plan 66

5.3.2 Experimental setup 66

5.3.3 Experimental results 68

6 Discussion and Conclusion **69**

6.1 Discussion 69

6.1.1 Performance comparison 69

6.1.2 Network expansion 74

6.1.3 Run time 76

6.2 Conclusion 77

6.3 Contributions 79

6.4 Future work 80

Appendices **87**

A Structured Literature Review **89**

A.1 Protocol 89
A.1.1 Defining the problem 89
A.1.2 Search terms 90
A.1.3 Complete search term 90
A.1.4 Inclusion criteria 91
A.1.5 Quality criteria 91
A.1.6 Selecting the final literature 92
A.2 Search engines and search strings 94
A.2.1 ACM Digital Library 95
A.2.2 ScienceDirect 95
A.2.3 CiteSeer 96
A.2.4 SpringerLink 97
A.2.5 IEEE Xplore 98
A.2.6 ISI Web of Knowledge 99
A.2.7 Google Scholar 100

B Input Data **103**

B.1 Mandl's Network 103

C Experimental Results **107**

C.1 Parameter settings results 107
C.2 Performance comparison results 109

Abbreviations

VRP	Vehicle Routing Problem
UTNDP	Urban Transit Network Design Problem
UTRP	Urban Transit Routing Problem
UTSP	Urban Transit Scheduling Problem
SI	Swarm Intelligence
ACO	Ant Colony Optimization
PSO	Particle Swarm Optimization
BCO	Bee Colony Optimization
CSS	Combined Swarm System
TOTFIT	Total Fitness Value
ATT	Average Total Travel Time

Chapter 1

Introduction

1.1 Background and motivation

Trondheim and neighboring municipalities are among the areas with the greatest population growth in Norway [Miljøpakken, 2014]. More people lead to more traffic, and without action, congestion and environmental problems in these urban areas will continue to increase every year.

Private transportation has many advantages for the citizens compared to the public ones, including a decreased travel time and a direct travel from the origin to the destination. However, private transportation has some negative issues. An increased number of vehicles on the roads, which further leads to increased traffic jams, air pollution, noise, and accidents are some of these negative issues.

Having efficient public transportation systems can substantially reduce the negative effects of private transportation. Public transportation systems are better suited for urban needs because they can transport more people per time unit than cars, and they need much less space. The environment package for transportation in Trondheim [Miljøpakken, 2014], aims to provide an improved public transportation system. With this effort, they hope to increase the number of public transportation passengers, and with this achieve lower emissions, decreased traffic jams, and less traffic noise. An inadequately designed transit network can cause few route opportunities and high travel times for routes with high demand, resulting in less people using the service. Therefore, public transportation systems should be improved by providing a more suitable transit network, in order to convince more people to use public transportation services instead of their private vehicles.

AtB [AtB, 2015] is responsible for planning and operating the transit network

throughout Sør-Trøndelag County in Norway, which includes the city of Trondheim. Bus services comprise the major component of the public transportation system in the county, and in Norway generally. Moreover, bus services have specific attractive features, such as flexible routes, low cost, easy implementation, flexible fleet size, and use of existing facilities. AtB's current solution consists of an experience based route network, where transport planners has constructed reasonable transit networks and schedules entirely manually, exploiting local knowledge. As a result, the efficiency of the resulting networks is dependent on the designers' experience and their existing knowledge of constraints and transportation demands.

In the past, transit planners have done a reasonable job designing transit networks and schedules without the assistance of scientific tools or systematic procedures. Nevertheless, for a large network it is almost impossible to develop an efficient transit route network and bus schedules relying only on experience and guidelines. This is because, in large urban areas the number of bus routes and bus stops is extremely large.

The problem of designing the optimal set of routes for a fleet of vehicles, to serve a given set of customers, is referred to the Vehicle Routing Problem (VRP). The Urban Transit Network Problem (UTNDP) is an example of this broad NP-hard optimization problem and involves designing urban transit routes and schedules. Because routing problems are represented as a road network by relevant locations in a graph, a graph database can be a natural way to represent the data. VRP and graph databases are described in depth in Section 2.2 on page 26 and Section 2.3 on page 27, respectively.

The manual attempts in providing acceptable solutions to VRPs are not able to solve these large network problems optimally. In order to overcome and contribute to this problem, the number of journal publications on VRPs has increased in recent decades. The increase in research on these areas is also due to the progress in computational resources, and this has opened new possibilities for modeling more complex routing problems. Swarm Intelligence (SI) has proven to solve a great number of NP-hard problems [Dorigo and Gabardella, 1997; Lucic and Teodorovic, 2003]. SI based algorithms are metaheuristic optimization algorithms and are typically used to find optimal solutions to combinatorial optimization problems. SI is described in depth in Section 2.1 on page 21. The employment of such an automatized method for generating bus routes, would not only release planning time for AtB, but could also increase the quality of the network.

1.2 Goal and research questions

This thesis will focus on the development of a swarm intelligence system to optimize bus routes in urban transit networks. We hope that this can result in an increased number of public transportation passengers, when the designed network hopefully

becomes more convenient for the passengers. This thesis will focus on creating beneficial routes with a low average travel time, as well as keeping the number of transfer per passenger at a minimum. This motivation is drawn to the following goal:

Goal: Develop a system to improve urban transit networks.

Based on the proposed goal, the following research questions are formulated:

RQ 1: What is the state-of-the-art in solving Vehicle Routing Problems using swarm intelligence methods and graph databases?
By answering this question, we will establish a theoretical foundation for the thesis, as well as identify methods proposed in published literature. We are in this thesis motivated in creating a solution that combines attributes from different swarm intelligence methods, and this question will help us establish if there have been any previous research attempting this. It will also help us establish if graph databases have been used in combination with vehicle routing problems and swarm intelligence methods in the past.

RQ 2: Is it efficient to add attributes from other swarm intelligence methods in order to improve a standard ant colony optimization implementation?
Ant colony optimization (ACO) algorithms have proven to solve NP-hard problems in the past. However, the algorithm has some well-known limitations, such as local convergence. By answering this question, we will establish if an ACO algorithm can benefit in additional attributes from other swarm intelligence methods.

RQ 3: Is it possible to apply the proposed system to optimize urban transit routes in large urban cities?
If the proposed system is to be used to optimize route networks in large urban cities, it must be able to support large networks with many bus stops, roads, and transit routes. This question aims to establish whether the proposed system is possible to use when a given transit network becomes significantly large.

1.3 Research methods

In this thesis, two research methods are applied. The first research method used is a structured literature review, introduced by [Kofod-Petersen, 2014]. This research is conducted in order to establish the state-of-the-art of using swarm intelligence methods and graph databases to solve Vehicle Routing Problems. The results of

the final synthesis are presented as the related work that further forms the basis for the proposed problem statement.

The second research method is the design and development of the proposed system. Experiments comparing the proposed system to a generic ACO algorithm and several published methods are conducted. To ensure a sufficient comparison basis, the proposed system use a well-known benchmark problem.

The proposed system also attempts to solve larger problems than the benchmark problem described above. By larger problems, we mean a network with a realistic number of bus stops, roads and routes in the route set. This will establish whether the proposed system supports larger networks, which further allow us to discuss the possible usability in a real urban city.

For all the experiments, numeric values of established performance criteria, including average travel time, are presented. These values are further used to discuss the performance of the proposed system

1.4 Thesis overview

Chapter 2 sets the context for this thesis. It starts by discussing the domain swarm intelligence in Section 2.1, and includes theory about ant colony optimization, bee colony optimization, and particle swarm optimization. In Section 2.2 the Vehicle Routing Problem is described, along with the Urban Transit Network Design Problem, which is a subproblem of the VRP. Section 2.3 describes the term graph theory and puts it in context with graph databases. The graph database Neo4j is also presented.

Chapter 3 describes the preparatory research conducted in order to define the problem to be solved in this thesis. Section 3.1 defines the research topic used in order to guide the structured literature review [Kofod-Petersen, 2014], which further is described in Section 3.2. Section 3.3 discusses and analyzes the related work and will eventually answer RQ 1. Finally, in Section 3.4, the problem statement is proposed based on the results presented in 3.3.

Chapter 4 describes the implemented system in detail. It starts with an overall description in Section 4.1, followed by a description of the chosen development environment in Section 4.2. Further, the assessments used in order to construct solutions are explained in Section 4.3. Sections 4.4 - 4.11 are detailed descriptions of each of the system's steps.

Chapter 5 documents the experiments conducted in this thesis. The experiments are divided into three parts, where each part contains the experimental plan, setup, and results. Section 5.1 documents the parameter setting experiment, and will

also document a justification of the selected parameters. Section 5.2 documents the performance comparison experiments with the ACO implementation and other published methods. Section 5.3 documents the network expansion experiments, which are conducted on larger networks.

Chapter 6 concludes this thesis. Section 6.1 evaluates and discusses the results obtained by the experiments in Chapter 5. Section 6.2 is the overall conclusion which also answers the goal and research questions sat for this thesis. Section 6.3 presents the contributions, and finally, Section 6.4 ends this thesis by looking at possible directions to take in the future.

Chapter 2

Theory and Background

2.1 Swarm intelligence

Swarm behavior is found in many different species in nature, including fish schools and bird flocks. Many species practice swarm behavior for a biological need to stay together, because predators usually attack one individual, and not an entire flock. Swarm behavior is also found in social insects like ants, wasps, bees, and termites. They collaborate on tasks for building nests, gathering food and organizing production. These social insect colonies have demonstrated that simple organisms can perform complex tasks by continuously interacting with each other. The colonies are highly distributed and self-organized, and they adapt well to changes in the environment. Swarm intelligence (SI) [Beni and Wang, 1989] is a branch of artificial intelligence that is strongly influenced by the swarm behavior found in the nature, attempting to adapt these characteristics in intelligent computer systems. SI based algorithms, such as the ones described in 2.1.1 - 2.1.3, are metaheuristic optimization algorithms. Metaheuristic algorithms are typically used to find good solutions to combinatorial optimization problems, but they do not guarantee an optimal solution.

2.1.1 Ant colony optimization

In nature, ants have proven to be extremely capable of finding an optimal or close to optimal route from the nest to a food source [Deneubourg et al., 1990]. Ants communicate by leaving a pheromone trail that other ants smell, and will follow by a certain probability. Most ant species leave a pheromone trail when returning to the nest from an important food source. The more pheromone units on a path

(i.e. the more ants that have chosen it), the greater the probability that other ants will choose it. The pheromone units on a path found early, but later discarded in favor of a better path, will evaporate. This results in a smaller probability to be selected, and thus ensures that better routes are favored.

In nature, ants are adaptable to changes, and manage to find the shortest path even though an obstacle is added to the current shortest path. This is illustrated in Figure 2.1. To the left, the ants have found an efficient route. In the middle, an obstacle is added, and the ants explore different paths. To the right the ants have found the most efficient route around the obstacle.

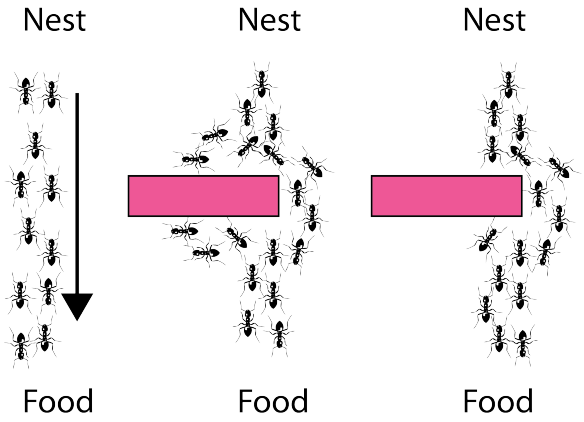

Figure 2.1: An illustration of how ants adapt to change

Ant colony optimization (ACO) is a class of graph representation based algorithms designed to optimize routing problems. The first description of an ACO algorithm, called Ant System (AS), was initially proposed by Dorigo et al. [1996]. The AS strategy developed by Dorigo tries to simulate the behavior of real ants, but unlike the ants in nature, pheromone trails are left by all ants. Algorithm 1 shows the metaheuristic of ACO, as described by Dorigo et al. [2006]:

Algorithm 1 The Ant Colony Optimization Metaheuristic

```
Set parameters, initialize pheromone trails
while termination condition not met do
    ConstructAntSolutions
    ApplyLocalSearch (optional)
    UpdatePheromones
end while
```

The idea of ACO algorithms is to create a decentralized system with multiple artificial ants. The artificial ants, like the one found in nature, influence each others decisions by using "pheromone". In the beginning, before any distinct pheromone trail is laid, the artificial ants' choices are random, and thus they perform a broad search in the environment. This randomness will decrease over time as the pheromone trails become more distinct. At the end of each iteration an amount of the pheromone evaporates. This reduces the probability of the artificial ants getting stuck at local optima.

Many versions of ACO algorithms have been proposed in the literature, and among the most successful are the Ant Colony System (ACS) proposed by Dorigo and Gabardella [1997] and the MAX-MIN Ant System (MMAS) proposed by Stützle and Hoos [2006]. The ACS implements both a *local* and a *global* pheromone update. The local pheromone update are applied by all artificial ants after each construction step, and the global pheromone update are applied by the evaluated best artificial ant at the end of each iteration. The MMAS only allows the evaluated best artificial ant to update the pheromone trail. The amount of pheromone given to the best path is determined, within a certain bound, by the quality of the solution found.

2.1.2 Bee colony optimization

As described in Lucic and Teodorovic [2003], bees are capable of performing a variety of complex tasks. Examples of these tasks are the collection and processing of nectar, which may be considered as very organized. The idea is that a bee that leaves the hive to gather nectar flies to the hives so-called "dance floor". The bees that already have found a good food source performs a "dance" at the dance floor to advertise that they have found a satisfying food source. Newly arriving bees will choose one of the dancers and follow it to the discovered food source. As stated in Lucic and Teodorovic [2003], the mechanism of deciding which bee to follow is not well understood. Nevertheless, it is considered that "the recruitment among bees is always a function of the quality of the food source". After a bee has gathered and returned the food to the hive, the bee has three options [Lucic and Teodorovic, 2003]:

1. Abandon the food source and return to the dance floor, and again follow a dancing bee
2. Continue to gather nectar from the food source without recruiting nest mates
3. Return to the dance floor and dance, and thus recruit nest mates before returning to the food source

Bee colony optimization (BCO) is a method that, like ACO, aims to create a decentralized optimization system with multiple agents, based on graph representation.

The idea is to apply the collective intelligence of the food gathering process to an optimization system. Like a typical ACO algorithm, a typical BCO algorithm is inspired by the way bees acts in nature, but where some features are added, and some are removed. Nikolic and Teodorovic [2014] describes a BCO algorithm where the artificial bee only has two options after returning from a food source: (1) abandon or (2) recruit. Lucic and Teodorovic [2003] gives the artificial bees attributes such as memory and perfect knowledge about the quantity of nectar collected by other artificial bees. These modifications are done in order to be more suitable for performing complex combinatorial problems.

2.1.3 Particle swarm optimization

As reported in Shi and Eberhart [1999], Particle Swarm Optimization (PSO) was first introduced by Eberhart and Kennedy in 1995. PSO is inspired by how the social behavior of flocks (such as flocks of birds) and schools (such as fish schools) cooperates. The idea is to update the individuals velocity according to the individual's experience and its companions' experience. This differs from other evolutionary computational algorithms, like Genetic algorithms, which uses evolutionary operators to manipulate the individuals. The basic concept is that each individual moves with a certain velocity and that this velocity is dynamically adjusted based on the experience. Each individual is a volume less particle (i.e. a point) in the D-dimensional search space. The ith particle position is represented as $X_i = (X_{i1}, X_{i2}...X_{iD})$. Each particle knows its own best position so far[1], represented as $P_i = (P_{i1}, P_{i2}...P_{iD})$, and the best position g, achieved among all the particles. These two positions are used to calculate the velocity, $V_i = (V_{i1}, V_{i2}...V_{iD})$, of the ith particle [Shi and Eberhart, 1999]:

$$V_{id} = w * V_{id} + c_1 * rand() * (P_{id} - X_{id}) + c_2 * Rand() * (P_{gd} - X_{id})$$

where w is a decreasing parameter called inertia weight added to PSO to balance local and global search, c_1 and c_2 are two positive constants, and *rand()* and *Rand()* are two random functions in the range [0,1]. Figure 2.2 illustrates how the particles in PSO explore in the early iterations of the algorithm, while figure 2.3 illustrates how the particles tend to act more organized and coordinated at the late iterations. The new position of the ith particle is calculated as follows [Shi and Eberhart, 1999]:

$$X_i = X_{id} + V_{id}$$

Because of the decreasing inertia weight, PSO may suffer from low global search ability at the end of the run, and thus risking getting stuck at a local optimum.

[1]The position that gave the highest fitness value

This may result in the algorithm failing to find the required optimum when the problem to be solved is very complicated and complex [Shi and Eberhart, 1999]. However, PSO has proven to create sufficient solutions to NP-hard problems including the Multi-Dimensional Knapsack Problem [Wan and Nolle, 2009] and the Urban Transit Routing Problem [Kechagiopoulos and Beligiannis, 2014].

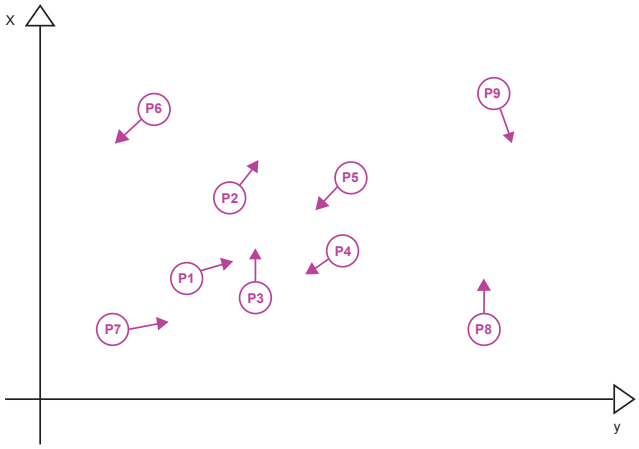

Figure 2.2: Illustration of particles in a 2D-space in an early iteration of the PSO algorithm (exploring)

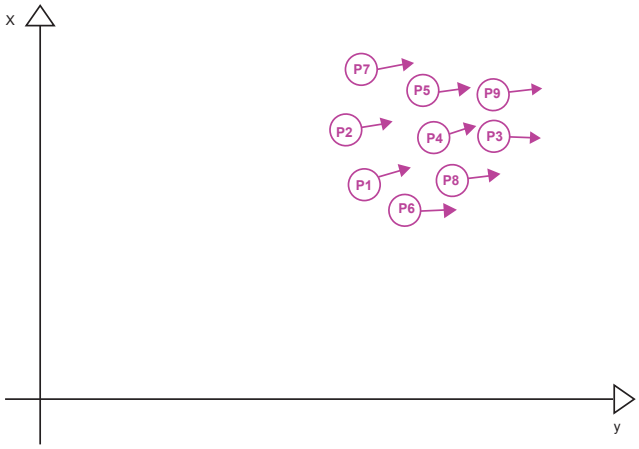

Figure 2.3: Illustration of particles in a 2D-space in a late iteration of the PSO algorithm (exploiting)

2.2 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) was first introduced by Dantzig and Ramser [1959] and is a generic name given to a broad class of optimization problems. It can be described as the task of designing the optimal set of routes for a fleet of vehicles to serve a given set of customers. The problem involves making deliveries to a set of customers with known demands on routes originating and terminating at one or more depots. The objective of any routing problem is usually the minimization of costs, such as reducing route lengths, number of vehicles, or minimize the total route cost.

Routing problems are represented as a road network by relevant locations in a graph. This graph will consist of a set of nodes and a set of edges, $G = (V, E)$. Nodes are directly connected by an edge, and the graph can be undirected or directed.

2.2.1 Urban Transit Network Design Problem

The problem of designing urban transit routes and schedules is called the Urban Transit Network Design Problem (UTNDP) and is a sub-problem of the VRP. The aim with this problem is to design efficient urban transit routes and schedules on an existing transit network while adhering to practical constraints. These constraints can include the maximum and minimum length for each route and the number of allowed routes in a route set. The two major components of UTNDP is called the Urban Transit Routing Problem (UTRP) and the Urban Transit Scheduling Problem (UTSP):

- UTRP is the task of developing a set of routes on an existing urban transit network, following certain constraints. It can be defined as the *physical* design of the UTNDP [Fan, 2009]. In a transit network, neighboring nodes (bus stops) are linked by an edge. Each step in a tour, traveling from one node to the next, is called a route, and a route will consist of several nodes connected by edges to form a path. Further, a route set consists of several routes combined. When all the routes in a route set are created, it will a form a route network. The selection of the best-generated route set is defined by an objective function. The goal is to find the optimal solution: The one with the best objective function value among all feasible solutions [Ferrucci, 2013]. A route network should include all the nodes, but may not contain all the edges present in the original transit network. The criteria for a good route set includes that the entire transit demand is served and that a large percentage of this demand is served through direct connections. In addition, the average travel time per transit user should be as low as possible.

- UTSP involves the development of schedules, arrival and department times, for the public vehicles, to travel along predefined routes. It can also be defined as the *operational* design of the UTNDP [Fan, 2009]. The contents of the schedule involves minimizing the time a passenger has to wait at each node (bus stop), following certain constraints, such as limited fleet size and bus capacity. The total waiting time includes the waiting time at their origin, in addition to the sum of the transferring time.

  UTRP and UTSP are usually implemented sequentially because the development of routes should be completed before the development of schedules.

There are some difficulties in solving the UTNDP. UTNDP is an NP-hard problem due to the need to search for optimal solutions among a large number of possible solutions. Some of the constraints may be difficult to model and satisfy. The generation and validation of the routes may involve a significant number of computations, which makes the run time high. Also, travel demand may be difficult to get hold of and are likely different from every hour of the day. The design will, therefore, be flawed if the data is of poor quality.

2.3 Graph databases

A graph database management system (graph database) [Robinson et al., 2013] is a database management system based on graph theory. The term graph theory has been used in centuries and was first introduced by the Swiss mathematician Leonard Euler (1707-1783). In 1736, he proved that there does not exist a closed walk that crossed exactly once each of the seven bridges of the river Pregel in Königsberg, now called Kaliningrad [Alexanderson, 2006]. Figure 2.4 on the next page shows Euler's original drawings from his paper written in 1736 [Euler, 1741] of the bridges in Königsberg.

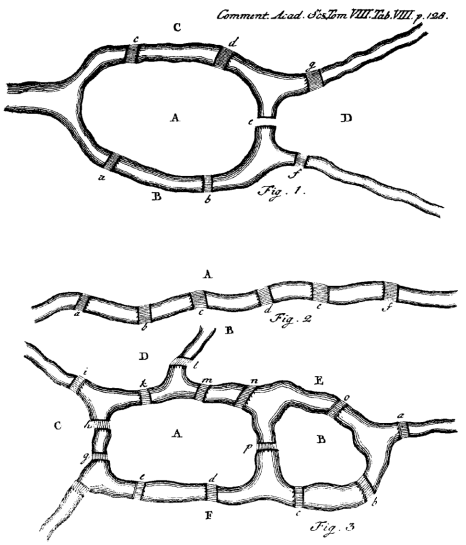

Figure 2.4: Eulers original drawing of the Seven Bridges of Köningsberg

Graph databases use graph structures for semantic queries with nodes, edges, and properties to represent and store data. The nodes represent entities, such as people, accounts, or bus stops, while the edges represent the relationships, such as "Friend of" or "Belongs to", between the nodes. A property is relevant information that can relate to either a node or an edge, such as "Name" or "Travel Time". Applications of graph databases can include calculating routes and finding the shortest path between locations in a network, such as a road or rail network, airspace networks, or logistical networks [Robinson et al., 2013, p.102].

2.3.1 Neo4j

Neo4j [Neo Technology, 2015] is an open-source graph database, implemented in Java. It is ranked the most popular graph database worldwide [DB-Engines, 2015] and is used by several large companies such as Telenor [ASA, 2015], Walmart [Walmart, 2015] and Cisco [Cisco, 2015]. It is a native graph database optimized and designed for storing and managing graphs and is known for extremely fast traversals of relationship. The underlying data model of Neo4j is the labeled property graph and is one of the most generic and versatile of all graph models [Robinson et al., 2013, p.73]. This graph data model gives four different fundamental building blocks to structure and store data. These building blocks includes Nodes to store entity information, Relationships to connect nodes to another, Properties for relevant information, and Labels for creating subgraphs.

A query that is extremely well suited for graph databases are queries for finding the paths between different nodes in a graph. Neo4j can be used to see whether a path exists, finding the optimal path, and looking for variability in the path [Bruggen, 2014, p. 51]. Neo4j includes built-in methods for finding the shortest path, including Dijkstra's algorithm. Dijkstra's algorithm [Cormen et al., 2009, p.658-662] maintains a set S of vertices' whose final shortest-path weights from the source s have already been determined. The algorithm repeatedly selects the vertex $u = V - S$ with the minimum shortest path estimate, adds u to S, and relaxes[2] all edges leaving u. The running time of Dijkstra's algorithm is $O((V + E)logV)$ and it is guaranteed to find the shortest path [Cormen et al., 2009, p. 661]. The Relationships in a Neo4j database can have different RelationshipTypes that, among others, enables the built-in Dijkstra's algorithm to find the shortest path using only specific RelationshipTypes.

[2]Making a change that reduces constraints.

Chapter 3

Preparatory Research

3.1 Refining the research topic

Cohen and Howe [1988] introduced a five-stage model for evaluating AI research in terms of a five-stage cycle, and the suggested model is used for evaluation throughout the project. The first stage in this model involves refining the research topic to a task and identifying a view of how to accomplish the task. As stated in the motivation, Section 1.1 on page 15, congestion and environmental problems in Trondheim is increasing every year due to the population growth, and efficient public transportation systems can reduce the negative effects issues. Since AtB's current solution consist of an experience based route network, it may not be the optimal solution. The task in this research is to optimize urban transit networks using swarm intelligence methods. Further is the goal to develop a system that improves urban transit networks, making them more convenient for passengers, and thus increase the number of public transportation passengers.

3.2 Structured literature review

A Structured Literature Review is a formal way of gathering available information from primary relevant studies [Kofod-Petersen, 2014; Kitchenham, 2007]. There are several advantages in using this model. These advantages include mapping out existing solutions, avoiding bias in the work, and highlighting areas where additional research is required. As proposed, a review protocol is developed. This protocol presents how each step is carried out, and the complete process is found in Appendix A on page 89. Table A.2 on page 94 shows the 11 final literature that were selected based on the protocol, which will form the foundation of our research.

The results of the final data synthesis is presented in the next section, all describing a vehicle routing problem attempted solved by a swarm inspired method. Reviewing this previous research will allow us to answer RQ 1, which is concerned about determining the state-of-the-art regarding swarm-intelligence, vehicle routing problems and graph databases. Further, the problem statement, based on the findings in the related work, will be presented in Section 3.4 on page 38.

3.3 Related work

3.3.1 Vehicle Routing Problems

As stated, all the retrieved literature from the structured literature review has studied the possibility of solving VRPs using swarm intelligence. Hsiao et al. [2004], Salehi-nezhad and Farrahi-Moghaddam [2007], Tripathi et al. [2009], Dias et al. [2014], Salehinejad and Talebi [2010], and Sedighpour et al. [2014] all use swarm intelligence to solve vehicle routing problems involving cars transporting either persons or goods.

Hsiao et al. [2004] presented an approach to search for the best path of a map considering the traffic loading conditions. To do this, they proposed an ACO algorithm to search for the shortest path from a desired origin to a desired destination. The presented algorithm is a classic ACO algorithm without changes and compared their algorithm to a brute method emphasizing on the time used to generate the route. Their results state that if the map consists of more than 200 nodes, the ACO performs better than a brute method. In fact, they found that the more nodes the map contains, the higher the benefit of using the ACO algorithm.

Salehi-nezhad and Farrahi-Moghaddam [2007] presented an ACO algorithm to search for the best path between two desired intersections in cities, called Ant-based Vehicle Navigation algorithm. To get more accurate results than a standard ACO algorithm they employed a method for rewarding and punishing the artificial ants. The path found by the "best ants" are given more pheromone than the path of the "bad ants". In order to find the best path, the presented algorithm is concerned about the parameters *distance*, *width*, *traffic load*, *road risk*, *road quality*, and *number of intersections*. The algorithm was applied on a part of the city of Kerman, and the results are encouraging and well described. The algorithm provides an easy method with low cost for vehicle navigation in cities without assisting GPS.

Tripathi et al. [2009] solved the vehicle routing problem with stochastic demand, in which the customer demand is modeled as a stochastic variable. They performed this using an improved version of ACO, called ns-AAA SO. The proposed algorithm orients the search progressively towards favoring the global optimal solution. To

do this they define that a complete iteration consists of two tours: The first tour is a social tour that corresponds to a standard ACO iteration. The second tour is a neighborhood tour where the ants are allowed to communicate important information found in the social tour and change their solutions. If the fitness value of the new solution is better, the old solution found in the social tour is replaced. To favor the optimal solution, the path of the global best path is given more pheromones. Further, to prevent the search from entrapment into a local optima, a minimum quantity of pheromone on any edge, t_{min}, is always maintained. The performance of ns-AAA SO was compared with both a standard ACO algorithm and a genetic algorithm. They found that ns-AAA SO performs better, regardless of the problem instance, than the other two algorithms compared to.

Dias et al. [2014] introduced an inverted ACO (IACO) algorithm. The idea is that the IACO algorithm inverts the logic of the classical ACO algorithm by converting the attraction of ants towards pheromones into a repulsion effect. The proposed approach was used in a decentralized traffic management system, where the drivers acted as the inverted ants. The drivers were repelled by the scent of pheromones (other drivers), and the system thus avoids congested roads. The described approach was compared to a shortest-time algorithm (ST), and the IACO algorithm performs better than the ST algorithm with the respect to trip time, travel length, fuel consumption and CO_2 emissions. This is as long as a considerable amount (25-50%) of the vehicles uses the inverted ant algorithm to decide which road to choose.

Salehinejad and Talebi [2010] introduced a route selection system that uses an Ant Colony System (AS), as described in Section 2.1.1 on page 21, to detect an optimum multiparameter direction between two desired points in urban areas. Their algorithm is called Fuzzy Logic-Ant Colony System (FLACS). FLACS differs from other ACSs by the employment of fuzzy logic for the local pheromone updating. The proposed system is concerned about the parameters *Distance*, *Traffic Flow*, and *Incident Risk* on each edge. FLACS also implements a tabu list, where visited nodes are added to avoid cycles. The system is applied to a part of London, United Kingdom, consisting of 42 junctions. Their proposed system is compared to a standard ACS and a A^*-ACS emphasizing on the parameters mentioned above. They found that FLACS performed better at average than both systems regarding operational cost, regardless of the importance in the parameters. It is worth mentioning, the estimation of further traffic data is done by artificial neural networks, and the traffic data for each system is not exactly the same.

Sedighpour et al. [2014] introduced a hybrid ACO (HACO) algorithm to solve the open vehicle routing problem (OVRP). This is a subproblem of the classical VRP where the vehicles are not required to return to the depot. To overcome some of the shortcomings of the original ACO, such as slow computing speed and local convergence, they made three improvements. First they equipped each node with a candidate list containing nodes nearby that had not yet been visited. Second, they applied several local search techniques to the n best solutions found each iteration

to improve them further. Third, they decided the amount of released pheromone based on the rank of the best-known solution found so far. The HACO algorithm was compared with three versions of PSO (standard PSO, PSO without one-point move (PSOWO) and PSO without neighbors (PSOWN) regarding performance. The algorithms were tested on fifteen different sets, consisting of 19 to 72 nodes with 2 to 7 vehicles fixed at the minimum possible. Their result table shows that HACO performs better than the others regardless of the test case used.

3.3.2 Urban Transit Network Design Problems

Urban Transit Network Design Problem (UTNDP), a subproblem of VRP, considers other objectives and requires other methods for generating solutions than classical VRP problems. As mentioned in Section 2.2 on page 26, UTNDP is divided into two parts. The first part includes creating urban transit routes on existing networks (UTRP) and the second part involves the development of schedules (UTSP). Yang and Yu [2007], Jiang et al. [2010], Poorzahedy and Safari [2011], Nikolic and Teodorovic [2014], and Kechagiopoulos and Beligiannis [2014] all describes solutions related UTNDP.

Yang and Yu [2007] presented an optimization algorithm for an urban bus network design (UBND), a problem closely related to the UTNDP. This algorithm is based on Dorigo et al. [1996]s Ant Colony Algorithm (ACA), called coarse-grain parallel ant colony algorithm (CPACA). CPACA is very similar to the original ACA, but it applies less communication between the ants by dividing the colony of ants into sub-colonies that runs in parallel and only communicate with each other. Their results are compared with the classical MAX-MIN ant system (MMAS) [Stutzle and Hoos, 1999] and with ACA with Ant-weight strategy (ACA+). They found that CPACA performed best regarding both average direct traveler density and run time.

Jiang et al. [2010] describes an improved ACO (IACO) algorithm to solve the UTRP. The specific improvement made to the algorithm is the implementation of a stagnation counter to determine whether the algorithm has fallen into stagnation. When there is no better solution found after an iteration, the stagnation will increase by 1. When the stagnation counter reaches a certain threshold, the pheromone levels associated with each edge is reinitialized. This improvement is done to compensate for the classical ACOs shortcomings of easily falling into stagnation and, therefore, obtain a local optimal solution. The IACO algorithm is, as the algorithm described by Yang and Yu [2007], compared to the classical MMAS. The results show significant improvement to the convergence speed compared to MMAS. The also found that IACO performed better both regarding average number of iterations and average path distance.

Poorzahedy and Safari [2011] proposed an Ant System for solving the bus network design problem (BNDP). BNDP is defined as the study of choosing a subset of interconnected bus routes among a given set of such routes. A successful solving of the BNDP, similar to UTNDP, minimizes the total travel time of the users of the network and the operational cost. Like the FLACS algorithm proposed by Salehinejad and Talebi [2010] and described in Section 3.3.1 on page 32, the proposed AS algorithm also employs a tabu list for each ant where visited nodes are added. Their solution generates multiple nests across the transit network and each nest is responsible for creating a subnetwork, which is combined to a complete bus network at the end. The system is only concerned about one objective; a combination of travel time for the users and the bus fleet size for the operator. The application was used to design the bus network of Mashhad and was further be compared with a genetic algorithm (GA). Their results show that their system performs better than the GA in both the number of routes, fleet size, in-vehicle travel time and waiting time. Both the GA and the AS performs significantly better than the existing solution on all measures.

Nikolic and Teodorovic [2014] proposed a method for solving the UTNDP. To do this, they used an improved version of the original BCO [Lucic and Teodorovic, 2003]. The bees start with an initial solution at each iteration, where the initial solution is the best-known solution so far. The initial solution is only updated if a better solution is found. The method was tested on Mandl's benchmark problem of a Swiss bus network [Mandl, 1980] and compared to competitive approaches (Mandl [1980]; Shih and Mahmassani [1994]; Baaj and Mahmassani [1995]; Bagloee and Ceder [2011]). The performance criteria used to measure the performance was regard to the percentage of total transfer demands satisfied directly (d_0), with one transfer (d_1), two transfers (d_2), or with more than two transfers or not satisfied at all (d_{unsat}). The methods are also compared regarding total in-vehicle travel time. The experiments are conducted on route set designs with four, six, seven and eight routes. They found that the proposed method performed best regarding total travel time and number of transfers if the order of importance was set to favor the passengers and the number of lines were greater than 4. If the order of importance was set to favor what was best for the operator, the method created the solution with the smallest fleet size independent of number of lines, otherwise the method performed mediocre regarding all the other measures.

Kechagiopoulos and Beligiannis [2014] designed and presented an original PSO algorithm without any changes or improvements. Their goal was to find an efficient solution to the UTRP. The target problem was, like Nikolic and Teodorovic [2014], Mandl's benchmark problem, and their algorithm was compared with competitive approaches, including genetic algorithms and other metaheuristic approaches mentioned in literature [Baaj and Mahmassani, 1991; Chakroborty and Wivedi, 2002; Kidwai, 1998; Fan and Mumford, 2010; Fan et al., 2009; Zhang et al., 2010; Chew and Lee, 2012]. The algorithms were compared to the same performance criteria and the same amount of route set sizes as Nikolic and Teodorovic [2014], but instead of comparing total in-vehicle time, they used the average in-vehicle time

experienced by each passenger (ATT). They found that the proposed algorithm performs better than the competitors regarding ATT independent the route size and achieves a better percentage of direct travelers (d_0) except when the route set size was four.

3.3.3 Discussion

Based on the proposed literature review we see that swarm intelligence inspired system has proven to be useful solving multiple vehicle routing problems. During the past decade, there has been published several research on the subject and many of these reports promising results.

We see that a lot of the recently published research addresses the weaknesses of classical SI-methods and makes changes to the original algorithms to overcome some of these weaknesses. In these cases we believe the conducted experiments should include comparison with the respective swarm method to indicate whether or not their solution improved the addressed shortcomings. Tripathi et al. [2009], Yang and Yu [2007], Salehinejad and Talebi [2010], and Jiang et al. [2010] all presented research where their swarm intelligence inspired algorithms were designed to overcome some of the known weaknesses. They compared their solutions with other corresponding swarm methods, achieving promising results. Because of this comparison, they can conclude whether or not the addressed weaknesses are improved. Sedighpour et al. [2014] improved the classic ACO to overcome slow computational speed and local convergence. However, they did not compare the proposed algorithm to other implementations of ACO, and the research will not be valid to conclude whether or not it actually overcomes some of ACO's weaknesses. Neither Dias et al. [2014] nor Poorzahedy and Safari [2011] tests their solution against other swarm intelligence methods, but against other reasonable algorithms, respectively a Shortest Time-algorithm and GA. The comparison against GA in Poorzahedy and Safari [2011] is relevant because they did not add additional features, other than a tabu list of all visited nodes, to the AS algorithm. In Dias et al. [2014]'s research, it makes sense to only test against a ST-algorithm, because they inverted the core factor of ACO. A comparison against a standard ACO would, therefore, be non-relevant. Salehi-nezhad and Farrahi-Moghaddam [2007] did not compare their algorithm against any other algorithm at all, which makes their results hard to verify.

The performance of metaheuristic methods, including swarm inspired methods, are highly dependent on their parameter settings. The process of parameter tuning is an important contribution to the field of swarm intelligence in general. Several researches, including Salehi-nezhad and Farrahi-Moghaddam [2007] and Yang and Yu [2007], describes their parameter setting as a product of "trial and error". We consider this to be a weakness of their research because it is not possible to validate their results. Sedighpour et al. [2014], Poorzahedy and Safari [2011],

and Kechagiopoulos and Beligiannis [2014] discussed and justified their parameter settings by conducting their experiments in two parts; one for parameter setting and one for performance. We consider this is a strength of their research.

The size of the test cases used is an important factor in determining both scalability and robustness of the proposed algorithm. Nikolic and Teodorovic [2014] and Kechagiopoulos and Beligiannis [2014] both uses Mandl's benchmark problem as input. This benchmark problem, presented in Fig. 3.1 on page 39, is a small network containing 15 nodes and 21 edges. Mandl's network is widely used and acknowledged by multiple researchers, including Baaj and Mahmassani [1991], Chakroborty and Wivedi [2002], and Fan [2009]. This is a strength of their research because it enables comparison of their results to a numerous of other solutions using the same benchmark problem. However, as mentioned, the Mandl network is quite small. The robustness of their algorithms regarding both time and space complexity could have been verified by also applying their algorithm to a larger test case. Salehi-nezhad and Farrahi-Moghaddam [2007] also applied their solution to a small test set, containing only 27 intersections and requiring only 5 ants and, like the authors that used the Mandl Network, the robustness could have been verified by also applying the algorithm to a larger test set.

3.3.4 Conclusion

Based on a review of the related work the first research question can be answered:

RQ 1: What is the state-of-the-art in solving vehicle routing problems using swarm intelligence methods and graph databases?

The structured literature review did not retrieve any previous research that used graph databases in combination with the vehicle routing problem and swarm intelligence. The state-of-the-art of solving vehicle routing problems using swarm intelligence methods can be summed up to being inspired by original SI-methods, but to add and remove features to make the methods more suited for the tasks. Ten out of the eleven reviewed papers made changes to the original method. We notice a trend of implementing a notion of the best known solution so far, and using this to either directly or indirectly improve the solutions created by the individuals of the swarm [Tripathi et al., 2009; Sedighpour et al., 2014; Nikolic and Teodorovic, 2014].

3.4 Problem statement

The goal of this research is to create a system to improve urban transit networks, which further hopefully will increase the number of public transportation passengers. As mentioned, the current solution of Trondheim' transit network consist of an experience based route network. This means the transit routes are not properly, computationally optimized concerning the travel demand and travel time. In order to contribute to this, we will in this thesis create a swarm intelligence inspired algorithm for the UTRP. A good route network will ensure that routes having the most traveling demands are satisfied with short paths and few vehicle transfers, making travel demand a key variable for the algorithm. AtB [AtB, 2015] does not possess accurate data about the travel demand, and detailed investigations into measuring and predicting travel demand is a complex research problem, beyond the scope of this thesis. Demand values and travel times are all provided for Mandl's benchmark problem [Mandl, 1979]. For the UTRP, Mandl's benchmark problem is widely used, whereas a recognized metric is established for evaluating the performance. Mandl's benchmark problem will therefore be used as the input data for most of the experiments in this thesis. The acknowledged performance criteria will be used to determine the performance of the algorithm and comparing the results with approaches published in the literature [Nikolic and Teodorovic, 2014; Kechagiopoulos and Beligiannis, 2014; Mandl, 1979; Kidwai, 1998; Fan and Mumford, 2010; Chakroborty and Wivedi, 2002; Zhang et al., 2010; Chew and Lee, 2012].

A standard ACO has several advantages for VRP, such as natural parallelism and continuous positive feedback, which allows good solutions to be identified fast. However, the standard ACO often has the shortcoming of getting stuck at a local optima. We will investigate the possibility of overcoming this drawback and to further enhance the optimization process by improving the standard ACO. Changes to the standard ACO has previously been done to overcome some of the algorithm's known weaknesses. Giving the ants knowledge of the visited nodes [Sedighpour et al., 2014; Salehinejad and Talebi, 2010; Poorzahedy and Safari, 2011], and adding a notion of the best-known solution so far [Tripathi et al., 2009; Sedighpour et al., 2014] showed improved performance. PSO and BCO use related approaches for rewarding the global best solution so far, and Kechagiopoulos and Beligiannis [2014] and Nikolic and Teodorovic [2014] demonstrates good performance with their respective PSO and BCO implementations. To answer RQ 2 on page 17, we will investigate whether incorporating additional features with respect to how PSO and BCO operate will improve the performance of the standard ACO.

RQ 3 on page 17 is concerned about whether the implementation can be used to optimize real urban transit networks. This cannot be fully answered until it is applied to a real city, but we will strive to create a method that is easily adaptable with the concerns of public transportation in cities in mind. Real urban cities are (often) a whole lot larger than Mandl's relatively small network (Fig. 3.1 on page

39). To test whether it is possible to apply the proposed algorithm on large transit networks, the algorithm will be implemented in a way that it is easily adaptable to various input values. Further it will be tested on large networks more similar to real transit networks.

As mentioned in Section 3.3.4 on page 37, we did not find any previous research that use graph databases in combination with the vehicle routing problem and swarm intelligence. The graph database Neo4j [Neo Technology, 2015] have several advantageous features for managing graphs, and we will in thesis explore how the usage of Neo4j affects the development process and the quality of the solution. In order to do this, the networks will be represented as Nodes and Relationships in a Neo4j graph database, and the generated routes will be further be added to the database. We will also strive to use the built-in methods in Neo4j in order to establish their usability in solving UTRPs with a swarm intelligence inspired system.

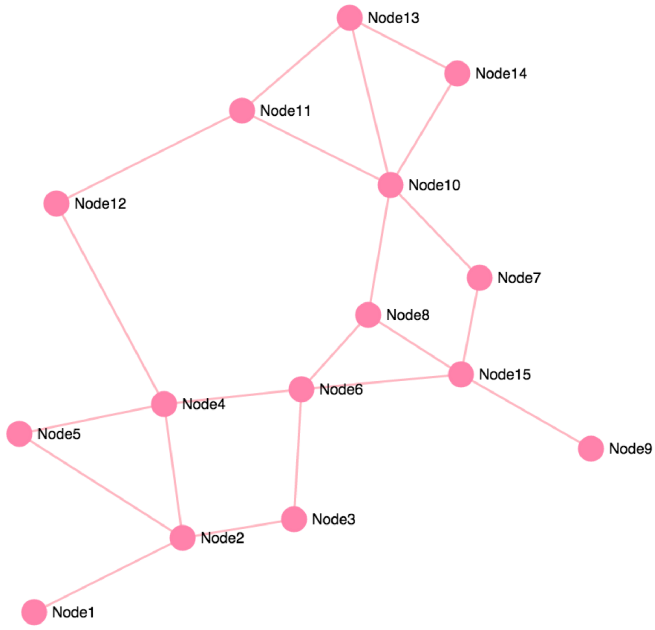

Figure 3.1: Illustration of Mandl's Network as a graph

Chapter 4

The Proposed System

4.1 Combined Swarm System

The basis of the proposed system, Combined Swarm System (CSS), is the ACO metaheuristic shown in Algorithm 1 on page 22. As initiated in Section 3.4 on page 38, some acknowledged attributes, inspired by PSO and BCO, will be added to the solution. The artificial ants generated, henceforth called ants, will also be given "memory". This attribute is given the ants to recall whether a node is visited in an earlier route within the same route set. This attribute is not linked to any optimization method from swarm intelligence. The feature is added because we observed by giving the ants memory, a higher amount of the generated route networks corresponded to a connected graph. This is one of the system's constraints initiated in Section 4.3.1 on the next page.

One of the supplementary attribute inspired by SI is the *Inertia Weight* from PSO. Inertia Weight is a decreasing parameter added to PSO for balancing local and global search. As illustrated in Figure 2.2 on page 25 and Figure 2.3 on page 25, the particle in PSO tends to explore more in early iterations, and becoming more organized and coordinated in the later iterations of the algorithm. In the initialization phase of CSS, an amount of the generated ants will be declared "crazy". A so-called "crazy ant", will work randomly, and not consider edge values when selecting nodes to be included in a route. The probability of an ant being declared "crazy" is given by a predefined start value (CA), decreasing iteratively with the inertia weight (IW).

The *following* attribute inspired by BCO is also added. As explained in 2.1.2 on page 23, deciding which bee to follow in BCO is considered to be a function of the quality of the food source found by the recruiter. After each iteration will

an amount of the evaluated best ants be "followed" in the next iteration. The "following ants" in the next iteration will follow the same path as the best ants from the previous iteration unconditionally.

4.2 Development environment

The proposed system is implemented using the Java programming language. In addition to being the programming language we are the most familiar with, Java is one of the most used programming languages worldwide. Henceforth, the possibility of further contributions will enhance. Further, Neo4j offers a rich set of integration possibilities for Java which are well documented. Nevertheless, compared to languages like C++, Java is often considered slow and memory-intensive [Alnaser et al., 2012]. However, Sestoft [2010] states that managed languages like Java are easier and safer to use than traditional languages like C++. Sestoft [2010] concludes, based on his conducted experiments, that "there is no obvious relation between the execution speeds of different software platforms, even for the very simple programs studied here: the C, C#, and Java platforms are variously fastest and slowest".

An embedded version of the Neo4j database is used to represent the network including nodes, edges and the created routes. An embedded database is preferred to lower the latency of the many reads and writes executed when running the system compared to a stand-alone version.

To handle project dependencies, such as Neo4j, and configuration details, such as how much memory that is allocated to the system, Apache Maven [Maven, 2015] is used. Maven is a tool that can be used for building and managing any Java-based project.

4.3 System assessments

The aim of designing a route network is to optimize specific criteria that define its efficiency and quality. Also, some real world constraints should be satisfied. The evaluation criteria and constraints used by the proposed system are inspired by [Kechagiopoulos and Beligiannis, 2014].

4.3.1 Constraints

During the route set generation, described in Section 4.8 on page 48, the system's constraints are the following:

1. **No cycles (or backtracks) in the graph is allowed.** In other words, a node visited once in a route should not be visited again in the same route. This corresponds to the real world constraint that a bus route should only include a given bus stop once.

2. **The route size is predefined. The routes shall not exceed the maximum or attain the minimum limit of nodes.** This corresponds to the real world constraint saying that a bus route should include at least a number n bus stops, and must not exceed a number m of bus stops. This number is usually sat by the bus service provider based the size of the transit network and cost limitations.

3. **The route set size is predefined.** This corresponds to the real world constraint that a service provider must be able to determine the number of bus routes in the transit network. This number is, like Constraint 2, sat based on the size of the transit network and cost limitations.

4. **The created route sets must correspond to a connected graph.** This corresponds to the real world constraint saying that a passenger must be able to travel between any two bus stops in the transit network.

4.3.2 Evaluation criteria

This evaluation of the ant's performance is done after each route set generation and is based on the following criteria:

1. The sum of the difference between the total travel time experienced by each passenger and the travel time of the shortest possible path, referred to as $F_1(r)$.

2. A score which reflects the percentage of passengers traveling either directly, transferring once or transferring twice, referred to as $F_2(r)$.

3. A score which reflects the percentage of passengers transferring more than twice, referred to as $F_3(r)$.

These evaluation criteria are used to calculate the Total Fitness, $TOTFIT$, of route set r, which process is described in Section 4.9.2 on page 49.

4.4 Initialization

4.4.1 Parameters

Nr.	**Parameter**	**Description**
1	s	The Colony Size
2	i	Number of iterations (the stop criteria)
3	E	Percentage of pheromones to evaporate at each iteration
4	p_v	The pheromone constant added to each edge, each time it is visited by an ant
5	p_b	Pheromone constant added to reward edges walked by following ants
6	AF	Percentage of ants to be followed
7	CA	The probability of a given ant to be declared "crazy"
8	RS	The number of routes in a complete route set
9	R_{max}	The maximum number of nodes in a route
10	R_{min}	The minimum number of nodes in a route
11	IW	The initial value for the Inertia Weight

Table 4.1: The parameters to be set for the system

The parameters of CSS are described in Table 4.1. The values of parameters 1-7, except parameter 4, are determined by a conducted experiment described in Chapter 5 on page 53. For comparison reasons, the value of parameters 8-10 will be the same as the parameters used in approaches described in the literature [Mandl, 1979; Kechagiopoulos and Beligiannis, 2014; Nikolic and Teodorovic, 2014; Kidwai, 1998; Fan and Mumford, 2010; Chakroborty and Wivedi, 2002; Zhang et al., 2010; Chew and Lee, 2012; Baaj and Mahmassani, 1991; Mumford, 2013]. The values of parameter 4 and 11 are constant at respectively 0.1 and 1.0.

The parameters E, CA and AF are all represented as percentages. E is stated as a percentage because the pheromone level changes over time. In the beginning, the level is quite small, and during the iterations the level increases. By stating E as a percentage, an equivalent amount of pheromone evaporates at each iteration. CA and AF are also stated as percentages to ensure that the quantity of both "crazy ants" (described in Section 4.5 on the next page) and "ants to be followed" (described in Section 4.10 on page 51) are the same regardless of the value of s.

4.4.2 Input data

The input data required includes:

- data concerning the structure of the road network (nodes with coordinates)
- data concerning the travel times between the nodes
- data concerning the travel demand between the nodes

Examples of such files can be found in Appendix B on page 103.

4.4.3 Network generation

The network consists of nodes and relationships, and these are represented in the graph database Neo4j. The nodes are created based on the files similar to the one presented in Table B.1 on page 104, which consist of the number of nodes and the coordinates for each node. These coordinates are used to represent the graph visually. An example of such a visual representation can be found in Figure 3.1 on page 39. The relationships between the nodes correspond to the travel time and demand between the nodes. These data are gathered from files similar to the ones presented in Table B.2 on page 104 and Table B.3 on page 105. Table B.2 shows the travel time between every two nodes in minutes. Some of the travel times are described as "Inf", meaning there is no direct link between the two nodes. Table B.3 shows the demand between every two nodes, which corresponds to the average number of passengers traveling between every two nodes each day. Both Table B.2 and B.3 are symmetrical matrices. Because the generated networks are directed, the travel time and demand between two nodes are the same in either direction.

4.5 Generating Combined Swarm colony

After the network initialization, a Combined Swarm colony is generated. Because the basis of the system is ACO, the individuals of the Combined Swarm colony will be referred to as ants. A colony consists of three types of individuals, and are the following:

- **Normal ants**: "normal ant" chooses the next edge to walk by a probability based on the edge value ev, described in Section 4.7 on the following page.
- **Following ants**: the "following ants", FA, will follow the same path as the best ants selected from the previous iteration. The amount of the ants to be followed in the next iteration, n, is determined by a percentage, AF, described in Section 4.10 on page 51. The same amount of FA will choose exactly the same nodes as the ant it is following.

  The FA will add additional pheromone to the edges walked by the best ants in the previous iteration, simply by walking the same path. To test if further

boosting is beneficial, an additional pheromone constant, p_b, will be added to the edges walked by the FA. The amount of p_b is selected after excessive testing. It is worth mentioning that in the very first iteration, there are no ants to be followed, and there will therefore neither be any FA.

- **Crazy ants**: By a given probability a normal ant is declared "crazy". A "crazy ant" chooses the next node at random, given the possible edges connected to the current node.

  The probability that a generated ant is declared crazy is partly determined by a predefined value, CA, achieved by the parameter testing, and partly determined by the inertia weight, IW. This probability, p, is calculated as follows:

  $$p = CA * IW$$

  As one can observe, the IW decreases at each iteration. This results in a smaller probability for the ants to be declared "crazy" in the later iterations. The rate in which the inertia weight decreases is dependent on the number of iterations, i. At each iteration IW is updated as follows:

  $$IW = IW - \frac{IW}{i}$$

  It might be worth mentioning that the FAs can not be declared crazy.

4.6 Selecting start node

At the creation of each route in each ant's route set, a start node is selected. This start node should ideally be selected randomly to allow for a variation in the routes created. We experienced, however, that if the start node is connected to another node that is only connected by one edge, the ant would often get stuck with only two nodes in the route set. Constraint 1 on page 43 specifies that no backtracking is allowed, and a node connected to only one edge will always be a start or an end node in a route. To prevent routes from containing only two nodes, nodes connected to a node with only one connecting edge are not selected as start nodes. Examples of such tabu nodes are node 2 and node 15 found in Figure 3.1 on page 39.

4.7 Selecting next nodes

How an ant selects the next node is dependent on which type of ant it is.

If ant a is a "crazy ant", the next node is chosen randomly based on the possible edges connected to the current node, without considering the edge value. If a is a "following ant", a consequently chooses the edges selected by the ant it is following.

If a is a "normal ant" the next node selection is based on an evaluated edge value, ev, for each of the possible edges, pe. pe is an edge that is connected to the current node and a node not yet visited in the given route. If no such edge exists, the given tour is terminated, else the value ev for each edge e in pe is calculated as follows:

$$ev_e = \frac{p_e}{\sum_{i=1}^{n} p_i} + \alpha$$

where p_e is the pheromone value of edge e, and n is the possible edges. α is a value sat to 1 if the connecting node is not yet visited in a's current route set, else it is sat to 0. This is due to the "memory" attribute initiated in Section 4.1 on page 41. ev_e is used to calculate the probability, $prop_e$, for e to be chosen, and is calculated as follows:

$$prob_e = \frac{ev_e}{\sum_{i=1}^{n} ev_i}$$

Each edge is given a range between 0 to 1 based on the calculated probability. Every real number between 0 and 1 is covered by a range exactly once. An edge with a high probability is given a broad range and vice versa. A random decimal number between 0 and 1 is used to determine which edge to choose. The edge that holds the random number in the range is chosen, and the connecting node of that edge is selected as the next node.

When an edge is selected, the pheromone value e_p of this edge is updated. This happens independently of ant type. e_p is updated as follows:

$$e_p \mathrel{+}= \frac{p_v}{T_e}$$

Where p_v is a predefined pheromone constant and T_e is the edge's travel time. As this formula shows, an edge with a shorter travel time is granted more pheromone than an edge with longer travel time. This is inspired by how Hsiao et al. [2004] updates pheromones, and designed to favor edges with lower cost, which, in this case, is lower travel times.

If a is "following ant", the chosen edge is granted additional pheromone, as mentioned in Section 4.5 on page 45. After the initial pheromone is added, e_p is further updated by the following formula:

$$e_p \mathrel{+}= \frac{p_b}{T_e}$$

where p_b is as mentioned a predefined pheromone constant determined after excessive testing.

4.8 Creating the route set

When a node is selected as the next node, it is added to the route, and declared as the current node. If the current route has reached its maximum number of nodes, R_{max}, the current route is added to the route set and a new route is initialized. As Constraint 3 on page 43 specifies, the route set size is predefined, and if the route set has reached its limit, RS, no new route will be created. If RS is reached, a has finished its tour, the route set is added to the Neo4j database and a is ready for evaluation.

In Neo4j, each route, r, produced by each ant, a, in each iteration, i, is given a RelationshipType, RT:

$$RT = i_{in}a_{an}r_{rn}$$

where in is the given iteration number, an is the ant number, and rn is the route number.

Neo4j's built-in Dijkstra Algorithm is used to find both the shortest possible path in the network and the shortest possible path using a given route set. The built-in Dijkstra is capable of taking one or more RTs as input, and finds the shortest path using only the given RT. If no such path exists, it returns null. The shortest possible paths are used in the evaluation phase described in the next section.

4.9 Evaluation

After each ant in the Combined Swarm colony has created a complete route set, the ant's route sets are evaluated. The route sets are evaluated as a whole because the connectivity and the paths chosen are dependent on the entire route set. The results of the evaluation determine whether a better route set than the best so far is achieved, in addition to determining which ants to be followed in the next iteration.

4.9.1 Removing ants that did not fulfill the constraints

Constraint 4 on page 43 specifies that a passenger should be able to travel from every node to every other node in the network. The first step in the evaluation is, therefore, to remove ants that have generated route sets which correspond to a

disconnected graph. For an undirected graph G to be classified as connected, there must be a path between every pair of nodes.

4.9.2 Calculating Total Fitness

In the next step, a fitness function, $TOTFIT(r)$, for the remaining ants' route sets are calculated. $TOTFIT(r)$ is used to compare and evaluate the solutions of the produced route sets. The calculation of $TOTFIT(r)$ for route set r is described as the sum of $F_1(r)$, $F_2(r)$ and $F_3(r)$:

$$TOTFIT(r) = F_1(r) + F_2(r) + F_3(r)$$

4.9.2.1 Calculating $F_1(r)$

$F_1(r)$ is referred to as the travel time experienced by each passenger compared to the shortest possible route in the network. For every passenger p, $F_1(r)$ is determined by the difference between the travel time of the shortest path given the route set, TT_{spr}, and the shortest possible path in the network, TT_{spn}:

$$F_1(r) = \frac{\sum_{i=1}^{p} TT_{spr_i} - TT_{spn_i}}{\sigma}$$

where σ is a positive user defined parameter used to control the importance of $F_1(r)$ compared to $F_2(r)$ and $F_3(r)$. In the presented contribution σ is sat to be ψ^2, where ψ is the number of nodes in the network. In Fan [2009], the author proposes two different methods for calculating TT_{spr} given a specific route set. In the first method, *Method 1*, the path with the shortest traveling time, not considering any transfer penalties, is chosen. In the second method, *Method 2*, the transfer penalties are considered, and the path with the shortest traveling time, including transfer penalties is chosen. To achieve the most accurate and realistic results *Method 2* is chosen for selecting r for p. This gives us the following equation for calculating TT_{spr} between two nodes:

$$TT_{spr} = IVT + (\gamma * NT)$$

where γ is the transfer penalty, and NT is the number of transfers. For comparison reasons, γ is sat to 5 minutes. IVT is the in-vehicle travel time and is equivalent to the sum of the travel times associated with each edge in the route. IVT is calculated using the built-in Dijkstra algorithm in Neo4j. The theoretical best value for $F_1(r)$ is 0, denoting no difference between TT_{spr} and TT_{spn}.

4.9.2.2 Calculating $F_2(r)$

$F_2(r)$ reflects the percentage of passengers traveling from their origin to their destination either directly, making a single transfer or transferring twice. Calculating F_2 is done using the following equation:

$$F_2(r) = (\tau * d_0(r)) + (\phi * d_1(r)) + (\omega * d_2(r))$$

where $d_0(r)$ is the percentage of passengers traveling directly, $d_1(r)$ is the percentage of passengers making a single transfer, and $d_2(r)$ is the percentage of passengers transferring twice. τ is sat to -3, ϕ is sat to -2 and ω is sat to -1. The size of τ, ϕ and ω is determined to favor route sets with many direct travelers over route sets with many one transfer travelers, and one transfer travelers over two transfer travelers. τ, ϕ and ω are all negative values because the smaller the values of both $F_1(r)$ (described in Section 4.9.2.1 on the preceding page) and $F_3(r)$ (described in Section 4.9.2.3), the better the route set. The theoretical best value of $F_2(r)$ is -300, denoting 100% of the passengers traveling directly from their origin to their destination. The theoretical worst value of $F_2(r)$ is 0, meaning all passengers transfers more than twice.

4.9.2.3 Calculating $F_3(r)$

$F_3(r)$ is a score that reflects the percentage of unsatisfied passengers. By unsatisfied passengers we mean passengers traveling from their origin to their destination, where the number of transfers is more than 2. $F_3(r)$ is calculated using the following formula:

$$F_3(r) = (100 - d_0(r) - d_1(r) - d_2(r)) * \beta$$

where, $d_0(r)$, $d_1(r)$, and $d_2(r)$ are the same values described in Section 4.9.2.2, and the number 100 represent 100%. β is a positive user defined parameter, which determines how much a given ant should be "punished" for creating a route set where one or more passengers must transfer more than twice. ψ is sat to 10, to be correspondent to the values of τ, ϕ and ω described in Section 4.9.2.2. The theoretical best value of $F_3(r)$ is 0, which reflects no passenger being unsatisfied. The theoretical worst value of $F_3(r)$ is 1000, meaning that all passengers are unsatisfied.

The sum of $F_1(r)$, $F_2(r)$, and $F_3(r)$ results in, as described above, the $TOTFIT(r)$ value of ant a. The lower the value of $TOTFIT(r)$, the better the route set r.

4.10 Selecting ants to be followed

After the $TOTFIT(r)$ value of each ant is calculated, the ants are added to a list sorted in descending order. The ant with the best $TOTFIT(r)$ value is placed in the first position and so on. As initiated in Section 4.5 on page 45, a percentage of the colony becomes followers and follows the same path as one of the best ants from the previous iteration. To determine the number of ants to be followed for the next iterations, a value NAF, is calculated:

$$NAF = ants_{size} * AF$$

where $ants_{size}$ is the number of ants that satisfied all constraints, described in Section 4.9.1 on page 48. The value of AF is selected after excessive testing described in 5.1 on page 53. Because the list is sorted with respect to the $TOTFIT(r)$ value, the NAF first ants in the list will be the NAF best ants. The NAF first ants will, therefore, get a follower in the next iteration.

4.11 Pheromone evaporation

To simulate how pheromone on paths in the nature evaporate, an amount of the pheromone on the edges will be removed after each iteration. The formula for updating the pheromone value e_p on edge e, according to evaporation, is as follows:

$$e_p \mathrel{-}= e_p * \frac{E}{100}$$

The processes described in Sections 4.5 - 4.11 combined, represents one iteration, and are all executed i times. The ants are recreated each iteration, but the ants initialized as "following ants" acts as clones of the best ants from the previous generation. The edges, including the pheromone level, are carried over to the next generation, along with the data from the global best ant so far. The system terminates when i iterations have finished running.

Chapter 5

Experiments and Results

5.1 Parameter settings

5.1.1 Experimental plan

Metaheuristics, like the ant colony optimization (ACO), requires good initial parameters to solve concrete problems optimally. The parameter settings experiment will study the effect of the variation of the parameters, and will be conducted in the attempt of finding the most optimal parameters for the system. As mentioned in Section 3.3 on page 32, several authors refer to their parameter settings experiments as a product of "trial and error", without presenting the parameter values tested. For contributing to the field and providing a starting point for future research, this thesis includes a complete review of the conducted experiment. Also, studying the effect of the additional parameters inspired by particle swarm optimization (PSO) and bee colony optimization (BCO) will help establish whether these attributes improves the standard ACO implementation. This is will further help answer Research Question RQ 2 on page 17.

5.1.2 Experimental setup

The parameters used in the proposed system (CSS) are described in Section 4.4 on page 44, and the parameters to be tested are presented in Table 5.1 on the following page. As one can see, each parameter is assigned a default value to be held constant while the other parameters are tested. Each parameter are run with minimum *four* different candidate values, presented in Table 5.2 on page 55. The selected value will be the one that produce the lowest Total Fitness ($TOTFIT$). As

stated in Section 4.9 on page 48, the lower the $TOTFIT$, the better the solution.

The default- and candidate values for parameters s and i are both inspired by the corresponding values described in related research [Salehi-nezhad and Farrahi-Moghaddam, 2007; Poorzahedy and Safari, 2011; Sedighpour et al., 2014; Kechagiopoulos and Beligiannis, 2014]. The parameters E, AF, and CA are considered unique for the proposed system, and their default values are chosen based on preliminary testing not included in this thesis. Even though the standard ACO implementations include an evaporation parameter, E is unique for this research because E is stated as a percentage. E, AF, and CA will be tested with values in the range from 0% to 100%.

The idea of the parameter p_b is to test whether rewarding edges in the best route sets, by adding more pheromone to edges walked by the "following ants", will boost the system's performance. The default value of p_b is 0.0, which implies that no extra pheromone is granted these edges. Because the amount of extra pheromone granted each edge is dependent on both p_b and AF, the candidate values of p_b will be tested with the selected value of AF. It is worth mentioning that the value of p_b must be seen in context with the value of parameter p_v. The value of p_v is the constant added to each edge each time it is visited by an ant. The values for p_v will not be tested with different values because the value could, in fact, be any real number, as long as it is constant. p_v is, as mentioned in Section 4.4 on page 44, sat to 0.1. Due to this, the candidate values of p_b will be in the range from 0.0 to 1.3.

The value of the inertia weight, IW, will not be tested with different values and is sat to 1.0. This is because, as described in Section 4.5 on page 45, IW directly affects CA, and because an implementation of different inertia weight strategies is beyond the scope of this thesis.

Parameter	**Description**	**Default Value**
s	The Colony Size	50
i	Number of iterations (the stop criteria)	50
E	Percentage of pheromones to evaporate at each iteration	10%
CA	The probability of a given ant to be declared "crazy"	10%
AF	Percentage of ants to be followed	10%
p_b	Pheromone constant added to reward edges walked by "Following Ants"	0.0

Table 5.1: Parameters to be tested, and their default value to be held constant while the other parameters are tested

For each candidate value of parameters s, i, E, and p_b, 30 runs will be carried out. For each candidate value of parameters AF and CA, 50 runs will be carried

out. The reason for the increased amount of runs is because the results of AF and CA will contribute in establishing RQ 2 on page 17. By running the system with additional runs, the margin of error will most likely decrease and will thus increase the validity of the results.

For all candidate values, the margin of error of the Confidence Interval with a confidence level of 95% will be presented. In addition will the best and worst produced $TOTFIT$ value and the standard deviation be presented. For each parameter, the value that resulted in the lowest average $TOTFIT$ will be selected.

All runs will be executed on Ubuntu instances provided by the Google Cloud Platform [Google, 2015]. The instances are of type "n1-highcpu-2", containing two 2.6GHz Intel Xeon E5 (Sandy Bridge) virtual CPUs with 1.8 GB memory.

5.1.3 Experimental results

Table 5.2 presents the parameters and candidate values tested, in addition the selected value for each parameter. The complete experimental steps with the corresponding results can be found in, Appendix C, Table C.1 on page 108 and Table C.2 on page 108.

Parameter	**Candidate values**	**Selected value**
s	10, 50, 100, 125	50
i	10, 50 , 100, 125	125
E	10%, 25% 50%, 90%	25%
CA	0%, 5%, 10%, 25%, 50%, 100%	25%
AF	0%, 5%, 10%, 25%, 50%, 100%	5%
p_b	0.0, 0.1, 0.5, 0.9, 1.3	1.3

Table 5.2: Results from the parameter settings experiment

5.1.4 Selecting final parameters

As one can observe in Table C.1 on page 108, increasing parameters s and i both decrease the $TOTFIT$ value. As mentioned in Section 4.9.2 on page 49, the smaller the $TOTFIT$ the better the solution. However, one aspect not considered in the initial parameter setting experiment is that the size of s and i affects each other. A colony of 50 ants with 125 iterations will produce close to similar results to 125 ants with 50 iterations, due to the fact that the total number of ants traversing the network will be approximately the same. The most optimal would, therefore, be to observe the results when increasing the value of these parameters together. However, an increase in the swarm size or number of iterations both clearly affect the computational cost of the proposed system. The average run time of each

run, with candidate values of 50 and 125, is presented in Table 5.3. As one can see, there is a difference in the running time with a colony size of 125 versus 125 iterations. Because the proposed system is going to be run an excessive amount of times when testing performance, increasing both of these parameter values will result in an increased run time. As seen in Table C.1, the value of 125 iterations produce better results than a colony size of 125. Due to the run time difference and the produced results, the selected value of parameter s is sat to *50* and i is sat to *125*.

Candidate value	s	i
50	181.9	187.7
100	883	756

Table 5.3: Running time in seconds for different candidate values of parameters s and i

Observing the results of different values of E in Table C.1 on page 108, evaporating 25% of the pheromone each iteration gave the best average $TOTFIT$, closely followed by 50%. The fact that a noticeable amount of pheromone evaporates each iteration seems to be beneficial for the system. Evaporation is important when it ensures that the pheromone level on routes explored in early iterations, but later discarded in favor of others, decreases. However, based on the presented results, removing over 90% at each iteration is a too excessive amount. The worst results were achieved when E was 10%. Evaporating only a small percentage of the pheromone may result in ants getting stuck at local optima because routes found in the early iterations will contain significantly more pheromone than newly discovered routes. By stating E as a percentage an equivalent amount of pheromone is removed from each edge, independent of whether the pheromone level is big or small.

Parameter CA produced best results with a value of 25%, meaning there is a 25% probability in that an ant is declared "crazy" at the beginning of each iteration. As mentioned, a "crazy ant" makes completely random decisions and select edges regardless of the pheromone value. CA will decrease at each iteration based on the inertia weight, as described in Section 4.5 on page 45. The results in Table C.2 on page 108 shows that the system benefits from the fact that some ants are declared crazy. As stated in Section 4.1 on page 41, if some ants are declared crazy, the probability of getting stuck at a local optima may decrease. However, if the value is greater than 25%, the average $TOTFIT$ results worsen. Not surprisingly, when half or more of the colony acts completely random, and the system looses some of the performing boosting features from ACO, such as favoring edges frequently walked by other ants.

The amount of AF determines the amount of "following ants" (FA) in the next iteration. An FA follow the same path as the best ants' paths unconditionally. Observing the results obtained in Table C.2 on page 108, the $TOTFIT$ value deteriorates when the amount of AF becomes greater than 25%. When the amount

of FA becomes too high, a relatively large number of normal ants will not be able to explore new (possibly better) routes in the next iterations. This may result in the system not being able to escape from a local optima. However, as one also can see, increasing the value computes better results than 0%. Rewarding some good route sets boosts the systems performance, and 5% is selected as the final parameter for AF.

The value of p_b is as mentioned strongly dependent on the value of AF. This is because the more following ants, the more pheromone added to each edge in the best route sets. The values of p_b was, therefore, tested with the selected value of AF. As one can observe in Table C.2 on page 108, the $TOTFIT$ value improves with a large amount of p_b. Granting the relatively small amount of edges selected by AF with extra pheromone, does improve the performance of the proposed system.

The computed Confidence Interval does not become remarkably better after running the proposed system 50 times, compared to 30 times. Produced results regarding parameters run 30 times can, therefore, be considered valid.

5.2 Performance comparison

5.2.1 Experimental plan

When the value of each parameter is selected, the comparison studies will determine the performance of the proposed system. The results produced by the proposed system will be compared to results published by Mandl [1979], Kechagiopoulos and Beligiannis [2014], Nikolic and Teodorovic [2014], Kidwai [1998], Fan and Mumford [2010], Chakroborty and Wivedi [2002], Zhang et al. [2010], Chew and Lee [2012], and Baaj and Mahmassani [1991]. Four different cases will be studied, each having different number of routes. For our experimental results to be straight comparable with the results published, cases with four, six, seven and eight routes in the route set will be examined.

To determine RQ 2 on page 17, which is concerned whether the additional attributes from swarm intelligence has improved a standard ACO algorithm, the proposed system will be compared to a standard ACO implementation. The standard ACO is identical to the proposed system, but without “following ants”, “crazy ants”, or “memory”.

5.2.2 Experimental setup

To determine the performance of the proposed system, some performance criteria will be used for evaluation. As stated in Kechagiopoulos and Beligiannis [2014],

the criteria were first proposed by Chakroborty and Wivedi [2002] to have a fair comparison between all system's results. The performance criteria are the following:

- $d_0(\%)$ - the percentage of passengers without any transfers.
- $d_1(\%)$ - the percentage passengers transferring once.
- $d_2(\%)$ - the percentage of passengers transferring twice.
- d_{unsat} $(\%)$ - the percentage of unsatisfied passengers. An unsatisfied passenger is described as a passenger with 3 or more transfers.
- ATT - the average travel time in minutes per transit user. In all approaches published in the literature, the transfer penalty of 5 min is applied to each route for each required transfer, and the same transfer penalty will be used by the proposed system.

The criteria for good performance includes that the percentage of passengers satisfied without any transfers is high and that the average travel time and the percentage of unsatisfied customers are low.

There will be carried out 50 runs per experiment, recording the values of d_0, d_1, d_2, d_{unsat}, and ATT of the best produced solution each run. The best-produced solution is the one with lowest $TOTFIT$ value. After 50 runs the best, average, median and worst solution for all runs will be presented along with the standard deviation.

The experiments will be run on Ubuntu instances provided by the Google Cloud Platform [Google, 2015]. For the experiments with four, six and seven routes, the instances used will be of type "n1-highcpu-2". These instances contain, as mentioned in Section 5.1.2 on page 53, two 2.6GHz Intel Xeon E5 (Sandy Bridge) virtual CPUs and 1.8 GB memory. For the experiment with eight routes an instance of type "n1-standard-2" will be used, due to the increased memory usage. This instance contains two 2.6GHz Intel Xeon E5 (Sandy Bridge) virtual CPUs and 7.5 GB memory.

5.2.3 Experimental results

Table 5.4 on the facing page presents the average produced results the proposed system (CSS) and the standard ACO implementation has produced. Best, worst, average, and median produced results in addition to the standard deviation can be found in Appendix C, Table C.3 on page 109.

System	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
ACO avg	81.92	16.13	1.86	0.09	10.43
CSS avg	85.21	13.49	1.30	0.00	10.27

Table 5.4: The best route set, having four routes, constructed by the generic ACO implementation and the proposed system.

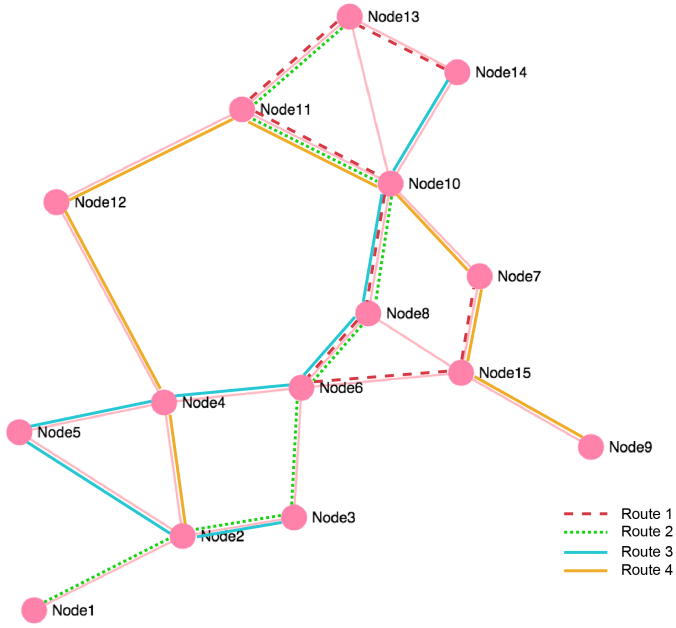

Figure 5.1: Illustration of the best route set on Mandl's Network, having four routes, constructed by the proposed system

Fig. 5.1 shows the representation of the best route set, having four routes, produced by CSS.

Table 5.5 on the next page presents the results produced by the proposed system, having four routes, and results from route sets constructed by other approaches. The published results are sorted in ascending order with respect to the ATT value.

System	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
Nikolic and Teodorovic [2014] avg	95.05	4.95	0.00	0.00	-[1]
Kechagiopoulos and Beligiannis [2014] best	91.84	7.64	0.51	0.00	10.64
Zhang et al. [2010]	91.46	8.54	0.00	0.00	10.65
Kechagiopoulos and Beligiannis [2014] avg	90.52	8.75	0.73	0.00	10.71
Chew and Lee [2012] best	93.71	6.29	0.00	0.00	10.82
Chew and Lee [2012] avg	92.88	6.91	0.20	0.00	11.16
Fan and Mumford [2010] best	93.26	6.74	0.00	0.00	11.37
Fan and Mumford [2010] SA[2] avg	92.48	7.52	0.00	0.00	11.55
Fan and Mumford [2010] HC[3] avg	91.83	8.17	0.00	0.00	11.69
Chakroborty and Wivedi [2002]	86.86	12.00	1.14	0.00	11.90
Kidwai [1998]	72.95	26.91	0.13	0.00	12.72
Mandl [1979]	69.94	29.93	0.13	0.00	12.90
CSS Best	87.73	10.98	1.28	0.00	10.03
CSS Average	85.21	13.49	1.30	0.00	10.27
CSS Median	85.81	13.29	1.09	0.00	10.26
CSS Worst	76.56	22.16	1.28	0.00	10.01
Standard Deviation	2.66	2.70	0.84	-	0.18

Table 5.5: Comparing the best route set, having four routes, produced by the proposed system, with route sets constructed by other approaches.

[1]: ATT not supplied
[2]: Simulated Annealing based system
[3]: Hill Climbing based system

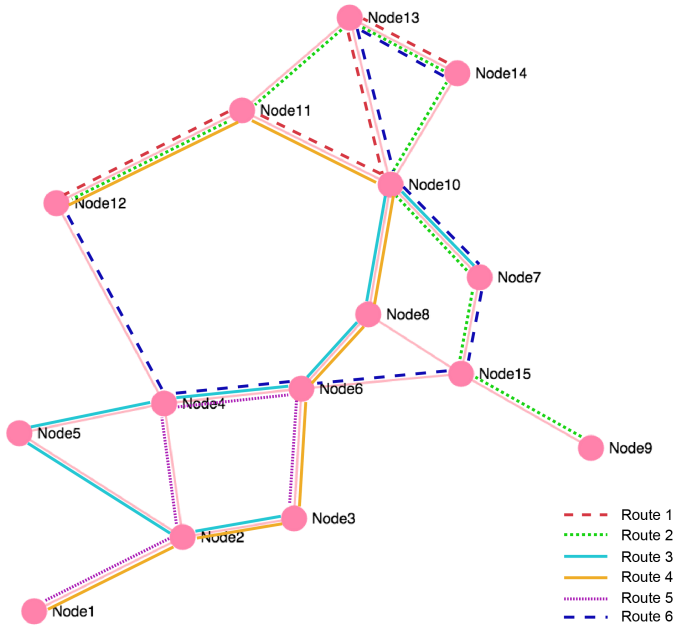

Figure 5.2: Illustration of the best route set on Mandl's Network, having six routes, constructed by the proposed system

Fig. 5.2 shows the representation of the best route set, having six routes, produced by CSS.

Table 5.6 on the next page presents the results produced by the proposed system, having six routes, and results from route sets constructed by other approaches. The published results are sorted in ascending order with respect to the *ATT* value.

System	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
Nikolic and Teodorovic [2014]	94.34	5.65	0.00	0.00	-
Kechagiopoulos and Beligiannis [2014] best	96.21	3.47	0.32	0.00	10.23
Kechagiopoulos and Beligiannis [2014] avg	95.62	4.28	0.10	0.00	10.28
Chew and Lee [2012] best	95.57	4.43	0.00	0.00	10.28
Chakroborty and Wivedi [2002]	86.04	13.96	0.00	0.00	10.30
Fan and Mumford [2010] best	91.52	8.48	0.00	0.00	10.48
Zhang et al. [2010]	91.12	8.88	0.00	0.00	10.50
Chew and Lee [2012] avg	93.85	5.88	0.24	0.03	10.51
Fan and Mumford [2010] SA avg	90.87	8.74	0.39	0.00	10.65
Fan and Mumford [2010] HA avg	90.23	9.26	0.51	0.00	11.69
Kidwai [1998]	77.92	19.62	2.40	0.00	10.78
Baaj and Mahmassani [1991]	78.61	21.39	0.00	0.00	11.86
CSS Best	89.53	9.25	1.22	0.00	10.03
CSS Average	87.17	12.0	0.82	0.00	10.11
CSS Median	87.93	10.98	0.77	0.00	10.03
CSS Worst	82.47	17.41	0.13	0.00	10.03
Standard Deviation	2.74	2.78	0.63	-	0.14

Table 5.6: Comparing the best route set on Mandl's Network, having six routes, produced by the proposed system with route sets constructed by other approaches.

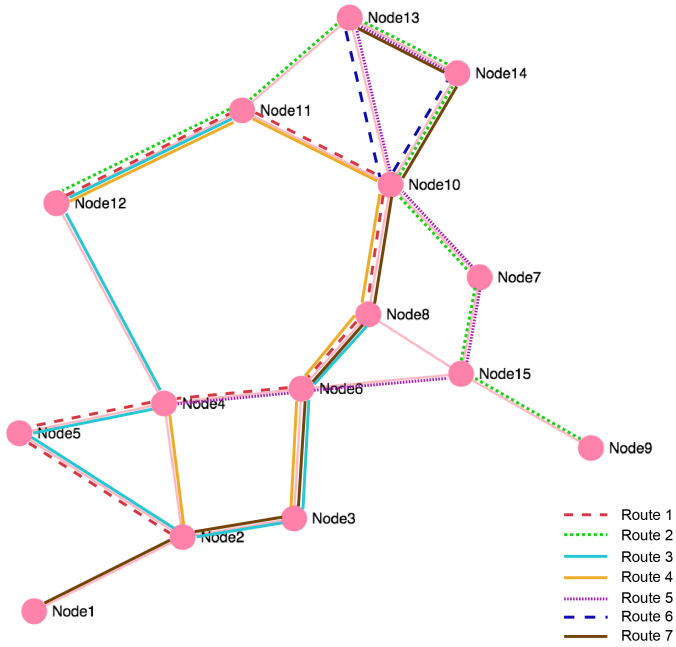

Figure 5.3: Illustration of the best route set on Mandl's Network, having seven routes, constructed by the proposed system

Fig. 5.3 shows the representation of the best route set, having seven routes, produced by CSS.

Table 5.7 on the next page presents the results produced by the proposed system, having six routes, and results from route sets constructed by other approaches. The published results are sorted in ascending order with respect to the *ATT* value.

System	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
Nikolic and Teodorovic [2014]	94.41	5.59	0.00	0.00	-
Chakroborty and Wivedi [2002]	89.15	10.85	0.00	0.00	10.15
Kechagiopoulos and Beligiannis [2014] best	97.17	2.83	0.00	0.00	10.16
Kechagiopoulos and Beligiannis [2014] avg	96.55	3.45	0.01	0.00	10.23
Chew and Lee [2012] best	95.57	4.42	0.00	0.00	10.27
Chew and Lee [2012] avg	96.47	3.53	0.00	0.00	10.31
Fan and Mumford [2010] best	93.32	7.13	0.32	0.00	10.42
Zhang et al. [2010]	92.89	7.11	0.00	0.00	10.46
Fan and Mumford [2010] SA avg	92.47	6.95	0.58	0.00	10.62
Kidwai [1998]	93.91	6.09	0.00	0.00	10.70
Fan and Mumford [2010] HC avg	92.21	7.13	0.66	0.00	10.74
Baaj and Mahmassani [1991]	80.99	19.01	0.00	0.00	12.50
CSS Best	89.85	8.67	1.48	0.00	10.03
CSS Average	88.49	10.72	0.79	0.00	10.08
CSS Median	88.12	10.92	0.90	0.00	10.03
CSS Worst	83.94	15.93	0.13	0.00	10.01
Standard Deviation	2.29	2.32	0.42	-	0.08

Table 5.7: Comparing the best route set on Mandl's Network, having seven routes, produced by the proposed system with route sets constructed by other approaches.

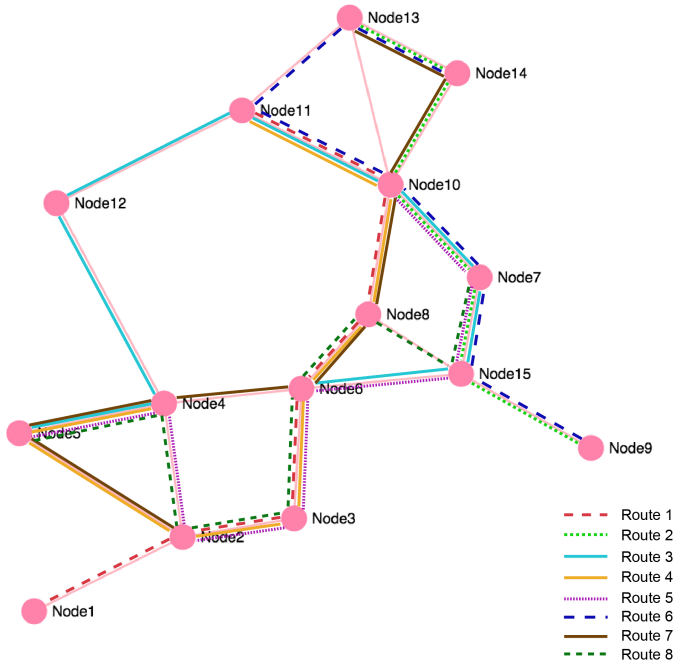

Figure 5.4: Illustration of the best route set on Mandl's Network, having eight routes, constructed by the proposed system

Fig. 5.4 shows the representation of the best route set, having eight routes, produced by CSS.

Table 5.8 on the next page presents the results produced by the proposed system, having six routes, and results from route sets constructed by other approaches. The published results are sorted in ascending order with respect to the *ATT* value.

System	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
Nikolic and Teodorovic [2014]	96.40	3.60	0.00	0.00	-
Kechagiopoulos and Beligiannis [2014] best	97.75	2.25	0.00	0.00	10.13
Kechagiopoulos and Beligiannis [2014] avg	97.47	2.53	0.00	0.00	10.17
Chew and Lee [2012] best	97.82	2.18	0.00	0.09	10.19
Chew and Lee [2012] avg	96.16	3.84	0.00	0.09	10.31
Fan and Mumford [2010] best	94.54	5.46	0.00	0.00	10.36
Zhang et al. [2010]	93.14	6.86	0.00	0.00	10.42
Chakroborty and Wivedi [2002]	90.38	9.58	0.00	0.00	10.46
Fan and Mumford [2010] Simulated Annealing	93.65	5.88	0.47	0.00	10.58
Fan and Mumford [2010] Hill Climbing	93.23	6.18	0.59	0.00	10.69
Kidwai [1998]	84.73	15.27	0.00	0.00	11.22
Baaj and Mahmassani [1991]	79.96	20.04	0.00	0.00	11.86
CSS Best	91.01	7.9	1.09	0.00	10.01
CSS Average	89.16	10.05	0.8	0.00	10.06
CSS Median	88.7	10.21	0.77	0.00	10.03
CSS Worst	92.42	6.81	0.77	0.00	10.08
Standard Deviation	2.24	2.14	0.68	-	0.07

Table 5.8: Comparing the best route set on Mandl's Network, having eight routes, produced by the proposed system with route sets constructed by other approaches.

5.3 Network expansion

5.3.1 Experimental plan

To determine RQ 3 on page 17, the proposed system will be tested on larger networks than the Mandl Network. This research question is concerned about whether or not it is possible to apply the proposed system to optimize transit networks in large urban cities. The majority of real cities (often) consist of larger transit networks than the relatively small Mandl Network. As an example, Mandl's transit network contains 15 nodes (bus stops), whereas the transit network of Trondheim municipality consist of 1289 bus stops (informed by Email correspondence with AtB [AtB, 2015]). The network expansion experiments will establish whether the proposed system supports larger networks as input, and if so, how these networks affect the run time and the result quality.

5.3.2 Experimental setup

To test whether the proposed system supports larger networks, the generated networks added as supplementary material to Mumford [2013] will be used for these

experiments. Coordinates for each node, travel time between connected nodes, and demand values between each two nodes are all provided in this supplementary material. In addition, the maximum number of nodes ($Max(n)$), minimum number of nodes ($Min(n)$), along with the size of the route set (RS_{size}) are all specified. The experiments will be run with the same parameter values as the selected parameter values described in Section 5.1 on page 53.

Network	**Nodes**	**Edges**	$Min(n)$	$Max(n)$	RS_{size}
Mumford0	30	90	2	15	12
Mumford1	70	210	10	30	15
Mumford2	110	385	10	22	56
Mumford3	127	425	12	25	60

Table 5.9: Networks with properties from the supplementary material from Mumford [2013].

Method 1 will be used to generate routes on these networks, and not *Method 2*, which is used when testing on Mandl's network. As mentioned in Section 4.9.2.1 on page 49, Method 1 selects the path with the shortest traveling time, not considering any transitions, and the transfer penalties are added after the route is selected. In Method 2, the transitions are considered, and the path with the shortest traveling time, including transfer penalties, is chosen. As Table 5.10 shows, Method 2 performs better. However, these experiments will be run in order to determine *whether* the proposed system supports larger input, and due to the excessive difference in run time, Method 1 is chosen for these experiments. The proposed system will only be run 10 times for each network, and the produced results concerning the performance criteria must, therefore, be considered as indicative.

Method	d_0	d_1	d_2	d_{unsat}	ATT	**Run time**
1	24.16	37.79	28.53	9.52	13.36	2334
2	40.29	46.38	12.96	0.37	13.17	37161

Table 5.10: Comparing the run time in seconds for Method 1 and Method 2 on the Mumford0 network.

The experiments will be run Ubuntu instances provided by the Google Cloud Platform [Google, 2015]. The instances used will be of type "n1-standard-2", which contains two 2.6GHz Intel Xeon E5 (Sandy Bridge) virtual CPUs and 7.5 GB memory.

5.3.3 Experimental results

Table 5.11 presents the average results concerning the performance criteria, produced by the proposed system.

Instance	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
Mumford0[1]	23.43	39.36	28.78	8.43	13.5
Standard Deviation	1.12	2.72	2.42	1.29	0.21
Mumford1[1]	11.03	24.2	30.16	34.61	20.78
Standard Deviation	0.52	1.09	0.89	0.68	0.55
Mumford2[2]	-	-	-	-	-
Standard Deviation	-	-	-	-	-
Mumford3[2]	-	-	-	-	-
Standard Deviation	-	-	-	-	-

Table 5.11: Results produced for the instances added as supplementary material of Mumford [2013]

[1]: Number of iterations, $i = 50$
[2]: Failed due to generation of more RelationshipTypes than allowed by Neo4j.

Chapter 6

Discussion and Conclusion

6.1 Discussion

6.1.1 Performance comparison

The best route set, having four routes, constructed by the proposed system (CSS), is illustrated in Fig. 5.1 on page 59. Comparison of the standard ACO implementation and CSS concerning the average produced results is presented in Table 5.4 on page 59. The best, worst, and median results along with the standard deviation can be found in Appendix C, Table C.3 on page 109.

As one can see in Table 5.4 on page 59, CSS performs on average better than the ACO implementation concerning all the proposed performance criteria. Observing Fig. 6.1, the ACO implementation performs on average worse than CSS already in the first iteration. One reason for this difference is that the ants in ACO implementations does not possess the "memory" attribute. This feature enables the ants to "remember" which nodes is already visited in the same route set. Adding this attribute makes the ants favor nodes not visited over nodes already visited, and thus increase the probability of all nodes within the route set being visited at least once. Constraint 4 on page 43 specifies that the route network must be connected, and without the added memory, ACO will produce less solutions satisfying this constraint. This again makes ACO produce more route sets that later will be discarded, and thus not evaluated.

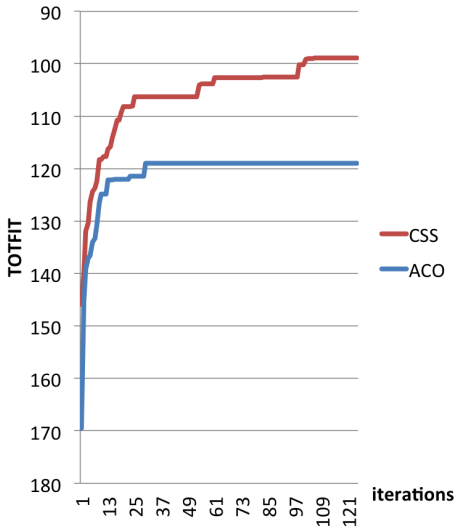

Figure 6.1: Average TOTFIT value of 10 runs at each iteration for ACO and CSS

Another reason for the difference in performance is because the standard ACO implementation has, as mentioned, a well-known shortcoming of entrapment in local optima. This disadvantage is demonstrated in Fig. 6.1. As one can observe, the ACO implementation manage to find good solutions fast. As described in Section 2.1.1 on page 21, the ants will perform a broad search in early iterations, due the lack of distinct pheromone trails. This randomness will decrease over time as the pheromone trails become more defined. Because pheromone evaporate over time, shorter paths will be favored over longer paths simply because shorter paths takes a shorter time. Observing Fig. 6.1, after approximately 35 iterations the amount of pheromone on the initial first best routes continue to increase. This will decrease the probability of new ants exploring possible better paths. The evaluation of the route set as a whole is done after each iteration, and this will determine how good the produced route set is. Because of the lack in rewarding the best route sets in the standard ACO implementation, there is no possibility for the ants to communicate information about good solutions.

As one also can see in Fig. 6.1, the proposed system manage to get out of this inconvenience, continuing to explore better solutions in the late iterations. Observed in the parameter settings experiment, extracted in Table 6.1 on the next page, the additional CA and AF parameters inspired by PSO and BCO, respectively, both improved the average $TOTFIT$ value.

The AF attribute is added to the proposed system to reward edges in the best

Parameter	CA	AF	p_b	$AVG(TOTFIT)$
CA	**0%**	10%	0.0	105.66
	25%	10%	0.0	**103.597**
AF (and p_b)	10%	**0%**	**0.0**	105.747
	10%	**5%**	**1.3**	**102.579**

Table 6.1: A selection of the average $TOTFIT$ with different parameter values for CA and AF

route sets with additional pheromone. This feature was partly inspired by Tripathi et al. [2009] and Sedighpour et al. [2014], who demonstrated that rewarding the best solution found so far improved their proposed solution. Rewarding the best solution may also be seen as the recruitment function in BCO. If an artificial bee in BCO has produced a good route set it can "recruit" other nest-mates and thus inform the others that a good route set is found. This process inspired to add the "following" feature to the proposed system. After the route sets are evaluated, an amount of the best ants with the best route sets is selected to be followed in the next iteration. The same amount of ants will follow the same routes and thus create the same route set. This will increase the pheromone units on the edges chosen by the best ants from the previous iteration, which further increases the probability of them being selected by other ants. Unlike the methods proposed by Tripathi et al. [2009] and Sedighpour et al. [2014] are we rewarding the n best solutions, instead of only the very best. Our system performed best with a relatively small amount of followers, but it also benefited from giving these edges additional pheromone (p_b). Rewarding edges in a large amount best route sets will result in over appreciating too many edges, which again will make it challenging to distinguish edges in *good* route sets from edges in the *best* route sets. By allowing some ants to be followers, it enables the ants to communicate good solutions with each other.

However, when the pheromone values on the best edges so far become too high, the probability of discovering new and possible better paths will decrease. The CA attribute was added to the proposed system to ensure more exploring throughout the iterations. CA denotes the amount of "crazy ants", which will explore edges random, regardless of the pheromone values. CA is inspired by how the particles in PSO explore solutions. In PSO, as mentioned in Section 2.1.3 on page 24, a decreasing parameter called the inertia weight balances local and global search. This makes the particles becoming more organized in the late iterations. Kechagiopoulos and Beligiannis [2014] showed that PSO can find promising solutions to the UTRP, and our CA attribute are partially inspired by this. To balance the global search by the "crazy ants" in the late iterations, the inertia weight (IW) inspired by PSO was added to the proposed system. The amount of "crazy ants" will decrease in line with IW, which again decreases in line with the number of iterations. Decreasing the inertia weight in PSO may result in low global search ability at the end of the run, and thus the possibly of getting stuck at a local optima. However, in the proposed system, the "crazy ants" are not searching towards

the best-known solution, and may thus prevent the same disadvantage.

To determine the solution quality of the proposed system, the system is compared to approaches published in the literature. In Table 5.5 on page 60, the results produced by CSS, having four routes, are compared with route sets published in the literature. As one can observe in Table 5.5 on page 60, d_{unsat} is 0, similar to all other approaches. The theoretical best value of for this criteria is 0, which reflects no passenger have to transfer more than two times. All approaches, except [Mandl, 1979; Kidwai, 1998; Chakroborty and Wivedi, 2002], perform better concerning the d_0, d_1 and d_2 criteria. However, the route set constructed by CSS produce a better ATT compared to route sets constructed by all other approaches. One reason for this is due to how the $TOTFIT$ value is calculated. The calculation of $TOTFIT$ is, as mentioned in Section 4.9.2 on page 49, the sum of $F1$, $F2$ and $F3$. As described in 4.9.2.1 on page 49, a weight parameter, σ, is used to control the importance of $F1$, $F2$, and $F3$. In the proposed system, σ is sat to favor $F1$, which is directly linked to a low ATT. Demonstrated in Fig. 6.2 on the next page, when traveling from Node 7 to Node 14, the system will choose to transfer from Route 4 to Route 3, which has a travel time of 20 minutes including a transfer penalty of 5 minutes. As we can see, there is a direct route between Node 7 and Node 14, but this route has a travel time of 27 minutes. As described in Section 4.9.2 on page 49, the route with shortest overall travel time will be chosen. The $F2$ parameter is concerned whether the proposed system has a high d_0, denoting a minimum number of transfers. If $F2$ was favored over $F1$, the route set shown in Fig. 6.2 on the next page would get a worse $TOTFIT$ and thus may not be considered as the best route set. It is worth mentioning that the proposed system will not select $F1$ unconditionally. As seen in the produced results, a high amount of d_0 is still an important factor when determining the best route set, but the selection of the best route set will be determined concerning the ratio between the two parameters. As mentioned in the motivation of this thesis, citizens often prefer private transportation because of the decreased travel time when no detours are needed. Then again, another important issue concerning passenger satisfiability, is not needing to change vehicles during a trip. Whether a passenger would travel direct with a larger travel time, versus transferring and thus decrease the travel time, is a matter of preferences. As one can see in all the approaches published in the literature including the proposed system, one will have to choose one at the expense of the other. One can argue back and forth about the importance of each criterion. We believe that in a modern urban city, a minimum travel time is the most important factor travelers. This is because some travelers may desire to get as fast as possible from their origin to their destination. We, therefore, choose to emphasize a short travel time. However, as mentioned, the produced route network also possess a relatively high amount of direct routes, giving many passengers opportunities to choose direct routes if it is desired.

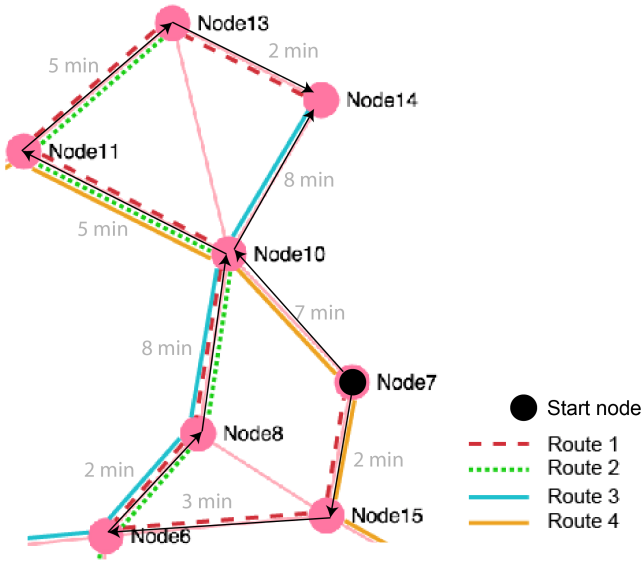

Figure 6.2: A fragment of the best route set, having four route sets, constructed by the proposed algorithm including transfer times in minutes between each node.

The best route set, having six routes, constructed by the proposed system is presented in Fig. 5.2 on page 61. The best route set, having seven routes, is presented in Fig. 5.3 on page 63. The best route set, having eight routes, is presented in Fig. 5.4 on page 65. The performance comparison for each route set size is found in Table 5.6 on page 62, Table 5.7 on page 64, and Table 5.8 on page 66, respectively. As one can observe, in all route set sizes the proposed system produce a lower ATT than all other approaches, and d_0 is below average, whereas d_{unsat} is still 0. As one can observe in Table 6.2, the amount of direct travelers increase, and the average travel time decrease in line with the number of routes. This corresponds to the growth in performance in all other approaches. The probability of finding direct routes and routes with small travel times will be greater as the number of routes increases.

Route Set	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
4	85.21	13.49	1.30	0.00	10.27
6	87.17	12.0	0.82	0.01	10.11
7	88.49	10.72	0.79	0.0	10.08
8	89.16	10.05	0.80	0.0	10.06

Table 6.2: Average produced values of the performance criteria for route set sizes four, five, six and seven on Mandl's Network

6.1.2 Network expansion

The results produced by the proposed system on the networks provided by Mumford [2013] is presented in Table 5.11 on page 68. As one can see, the value of parameter i is sat to 50 when testing the networks Mumford0 and Mumford1. While testing Mumford0 and Mumford1 with the original value for i, the proposed system did not manage to produce any results. The reason for this is because the number of generated RelationshipTypes exceeds the limit of RelationshipTypes sat by Neo4j, which is $2^{16} = 65536$ RelationshipTypes. All of the Mumford networks have an increased number of routes in the route sets (NRS) compared to route set sizes tested on the Mandl Network. As mentioned in Section 4.4.3 on page 45, to distinct the routes from each other, each route in each ant's route set in each iteration has an unique RelationshipType. The number of RelationshipTypes (NRT) is, therefore, dependent on both the number ants, the number of iterations and the NRS. As demonstrated in Table 6.3 on the next page, neither of the Mumford-networks is within this limit with the original value for i. However, by reducing i to 50, both Mumford0 and Mumford1 was within the limit. As demonstrated in Section 5.1.3 on page 55, the higher the i, the better the result. Due to the reduction of i, the results of the experiments run on the Mumford0 and Mumford1 networks have suffered. To be able to run experiments on Mumford2 or Mumford3, either s or i would have to be further reduced. Because further reduction of both these parameters will lead to a further deterioration of the results, this was not carried out.

As explained in Section 3.4 on page 38, we wanted to investigate how the usage of Neo4j would affect our proposed solution. To accomplish this, we included some artifacts from Neo4j, including the built-in Dijkstra. This built-in method can, as mentioned in 2.3.1 on page 28, find all paths in addition to finding the shortest possible path between two nodes. The method also includes the possibility of finding the shortest path within one or more RelationshipTypes. This function makes it possible to evaluate unique route sets created by unique ants. By adopting this built-in method, the development time of the proposed solution was reduced. It is also sufficient to apply this method on small networks, like the Mandl Network. However, when the network size increases, an inadequate amount of RelationshipTypes are required to be generated.

Comparing the results produced by the proposed system on the Mumford0 and Mumford1 networks to the results on the Mandl Network, one can see that the performance regarding both the number of direct travelers and unsatisfied travelers have worsened.

First of all, it is worth mentioning that the Mandl Network, the Mumford0 Network and the Mumford1 Network are three different networks, and may not be entirely comparable. However, given the increased route set sizes of the Mumford-networks, the experiments should be able to produce similar results regarding direct travelers and unsatisfied travelers as the experiments ran on the Mandl Network.

A reason for the the inferior results, may be the reduction of the value of i. The Mumford networks, which are twice or more the size than the Mandl Network, will be explored by 3750 fewer ants in total. This results in a decreased probability of finding the best route sets.

A second reason is, as mentioned in Section 5.3.2 on page 66, that Method 1 is employed when selecting routes, whereas Method 2 is used in the experiments run on the Mandl Network. In Method 1, the transfer penalties are not considered when selecting a path, which will increase the probability of selecting paths with many transfers. This again leads to an increased number of unsatisfied passengers, and a decreased number of direct travelers compared to using Method 2. The average travel time will also increase because transfer penalties are added to the total travel time afterward.

NRS	NRT^1
4	25000
6	37500
7	43750
8	50000
12	75000
15	93750
56	350000
60	375000

Table 6.3: Number of RelationshipTypes that will be generated using different route set sizes, with 125 iterations and a swarm size of 50.

[1]: 50 (s) * 125 (i) * NRS

Finally, when calculating the $TOTFIT$ value, as described in Section 4.9 on page 48, $F_1(r)$ is emphasized more than both $F_2(r)$ and $F_3(r)$. As described in Section 6.1.1 on page 69, $F_1(r)$ is already favored when running the tests on the Mandl Network, which resulted in the best average travel time over all the published systems compared to. When the networks becomes significantly enlarged, such as the networks provided by Mumford [2013], the value of $F_1(r)$ overwhelms the values $F_2(r)$ and $F_3(r)$. The effect of $F_2(r)$ and $F_3(r)$ will thus border against zero. In Table 6.4 the results of the average $TOTFIT$ is shown for each of the test cases. As one can observe, the average $TOTFIT$ is much larger for the Mumford0 and Mumford1 cases compared to the Mandl cases. The value of $F_2(r)$ is, as stated in Section 4.9.2 on page 49, between -300 and 0, and the value of $F_3(r)$ is between 0 and 1000. The value of $F_1(r)$ is dependent on network size and the total demand, and as Table 6.4 on the next page demonstrated, dividing $F_1(r)$ on the $nodeSize^2$ is not sufficient for these network sizes.

Number	Network	Average $TOTFIT$
1	Mandl (4 routes)	98.0
2	Mandl (6 routes)	89.5
3	Mandl (7 routes)	87.1
4	Mandl (8 routes)	83.4
5	Mumford0	3886.4
6	Mumford1	14459.8

Table 6.4: Average Total Fitness for all tests cases. 1-4 are the results of 50 runs, 5-6 are the results of 10 runs.

6.1.3 Run time

The run time for the proposed system is dependent on the NRS, the number of ants (s), the number of iterations (i), the size of the network, and whether Method 1 or Method 2 is used. Of course, the run time is dependent on both the s and i because both parameters increase the number of route sets to be generated and evaluated. Because a route set must be evaluated on the shortest travel time between every two nodes in the network, the run time increases as the network expands. As one can observe from Table 5.10 on page 67, the run time is highly dependent on whether Method 1 or Method 2 is used. The run time is, in fact, more than ten times greater when using Method 2 compared to Method 1. In Method 1, Dijkstra only will only need to find the absolute shortest path between two nodes, while in Method 2 Dijkstra will need to find all possible paths between the two nodes to further evaluate them based on the travel time and potential transfer penalties. The difference in using Method 1 and Method 2 will increase as both the network and the NRS increases, due to the enlarged number of possible routes. As one can see from the average run times regarding the experiments using the Mandl Network in Table 6.6 on the next page, the run time increases drastically from 7 to 8 routes. This is may be because, in the generation of 8 routes, the need for more memory led to using an instance from Google with more memory and less CPU power.

Network	Run time[1]
Mumford0	2368.0
Mumford1	5862.4

Table 6.5: Average run time in seconds of 10 runs using the Mumford Networks

[1]: Swarm size, $s = 50$, Number of iterations, $i = 50$

Network	Run time[1]
Mandl (4 routes)	673.0
Mandl (6 routes)	2442.1
Mandl (7 routes)	3891.1
Mandl (8 routes)	8515.9

Table 6.6: Average run time in seconds of 50 runs for each route set size on the Mandl Network

[1]: Swarm size, $s = 50$, Number of iterations, $i = 125$

6.2 Conclusion

We have in this thesis demonstrated how swarm intelligence inspired methods can create sufficient solutions to Urban Transit Routing Problems (UTRP). This is managed by conducting a structured literature review [Kofod-Petersen, 2014], as well as designing, implementing and excessively testing the proposed system. These processes are performed in order to establish the formulated Research Questions, introduced in Section 1.2 on page 16. RQ 1 is answered in Section 3.3.4 on page 37, after a thorough analysis of the primary relevant studies.

The conducted experiments and a discussion of the obtained results helped establish RQ 2 and RQ 3, which both are answered below.

RQ 2: Is it efficient to add attributes from other swarm intelligence methods in order to improve a standard ant colony optimization implementation?

The proposed system, Combined Swarm System (CSS), is compared against a standard ACO implementation to establish if the additional attributes added from bee colony optimization and particle swarm optimization were effective. The proposed ACO implementation was identical to the proposed system, but without the additional "memory", "following ants", and "crazy ants" attributes. This resemblance enabled a direct comparison of the two, and a viable comparison basis. The obtained results demonstrate that the proposed system on average performs better than the standard ACO implementation regarding all performance criteria.

The computational results of the proposed system were also compared with eight other methods published in the literature. CSS shows promising and competitive results, especially regarding the average travel time experienced by travelers. The results are all compared on the basis of Mandl's benchmark problem of a Swiss bus

network, which is a well-known and accepted benchmark problem.

Based on the comparison of the standard ACO implementation and CSS we cannot, unambiguously, conclude that the additional attributes inspired by PSO and BCO were the only reason for the improved performance. The additional "memory" attribute, inspired by Dorigo et al. [1996]; Sedighpour et al. [2014]; Poorzahedy and Safari [2011]; Salehinejad and Talebi [2010], was also implemented in the proposed system and is partly responsible for the performance improvement. However, results from the individual tests conducted on the additional parameters inspired by PSO and BCO demonstrated further improvement. With this we can, therefore, conclude that the additional attributes inspired by other swarm intelligence methods improved the standard implementation of ACO.

RQ 3: Is it possible to apply the proposed algorithm to optimize urban transit routes in large urban cities?

We have conducted additional experiments on larger networks, more similar to real transit networks. This is because the Mandl's Network used for comparison is a relatively small network. We have also investigated how the usage of a Neo4j graph database affects our development process and the quality of the solution.

Neo4j includes several features that are advantageous for these types of routing problems. However, the proposed solution as-is will not be possible to use for optimizing transit routes in large urban cities. This is because the number of RelationshipTypes generated by the proposed system generally will be above the limit of allowed RelationshipTypes in Neo4j. The number of RelationshipTypes generated are dependent on the swarm size, number of iterations and allowed routes. As an example, Trondheim consists of 42 routes compared to the 4-8 routes in the Mandl Network. If we were to use the current solution to optimize the route network in Trondheim, the number of RelationshipTypes created would be 8056250 whereas the number of allowed RelationshipTypes in Neo4j is 65536. To use the proposed system on larger networks, changes in the utilization or removal of Neo4j will have to be done.

The run time of the proposed system is dependent on the method used for evaluation, as well as the size of the network, number of iterations and colony size. However, generating urban transit routes are not a frequent task. Changing the transit routes often will result in unsatisfied passengers due to the frequent need to adapt to changes. Moreover, once an optimal urban transit network is created, there will be no need for frequent changes. Because of this, the runtime will not be an issue, given that the system creates a better solution than the one that already exists.

Overall conclusion

The goal of this thesis has been to develop a system in which improves urban transit networks. The motivation is that the improved transit routes further can increase the number of public transportation passengers. With a sufficient transit network, public transportation will be more attractive to urban travelers.

A system for creating urban transit networks is developed. The proposed system demonstrates good performance, especially regarding the average travel time per transit user. The proposed system as-is is not possible to apply on large networks, which a majority of real transit networks are, due to the described limitations of the Neo4j implementation.

6.3 Contributions

In this thesis, we proposed a system for the urban transit routing problem. The proposed Combined Swarm System creates feasible and efficient route networks with Mandl's benchmark problem [Mandl, 1979] as a basis. The system shows especially promising results concerning the average travel time per transit user.

We have demonstrated that the performance of a standard ant colony optimization algorithm improves when adding additional attributes inspired by other swarm intelligence methods.

We also conducted experiments regarding the parameter setting. The parameter settings have a great influence on the performance of metaheuristic methods, like the proposed system. By describing the experiments conducted and justifying the choices regarding the parameter setting, we contributed with a valuable starting point for future research.

The implemented system also utilize the use of the graph database Neo4j. Neo4j has several advantages, such as storing the objects as a graph and providing built-in graph algorithms. However, we have demonstrated some shortcomings of Neo4j for the proposed system. Both the advantages and shortcomings of Neo4j are considered as a contribution to the field and future research.

We contributed with the results of a conducted Structured Literature Review. The retrieved literature can provide important information to researchers who attempt to solve vehicle routing problems with swarm intelligence in the future.

6.4 Future work

Use Neo4j differently or remove from implementation

The proposed system does not perform sufficiently when the network size is bigger than the Mandl Network. Supporting larger networks is vital, because most real world transit networks are significantly larger than the Mandl Network. A reason for the insufficient performance is that the current solution with Neo4j does not support the excessive amount of RelationshipTypes required when the network size increases. An implementation where the generated amount of RelationshipTypes were significantly smaller would have been an interesting approach. Neo4j's built-in method is used for the traversal of graphs and is dependent on RelationshipTypes to distinguish the generated routes. Adding the "properties" function to edges instead of RelationshipTypes would be one approach to change the usage of Neo4j. However, an implementation of Dijkstra's algorithm from scratch will then be required.

The run time of the proposed system is relatively large. The run time increases fast if either the size of the network, number of iterations, the size of the colony or number of routes in a route set increases. We believe the use of Neo4j is at least partially responsible for this increase in run time. An implementation of the current system without Neo4j would, therefore, be interesting in order to investigate an possible change in run time.

Adjust the Total Fitness Function

The Total Fitness Function used for evaluation in the proposed system is sat to favor a low traveling time over the importance of number of transfers. This resulted in the lowest average travel time of all the compared researches. However, this favoring seems to increase as the network size increases. The influence of number of direct transfers borders thus against zero, meaning the number of direct transfers is not taken into consideration in the evaluation phase. A satisfied passenger emphasizes both a minimum travel time *and* the a minimum number of transfers. However, one would often have to choose one at the expense of the other. An interesting approach would be to create a Total Fitness Function that easily allows a service provider to determine the importance of the two.

Use the proposed system to solve the Urban Transit Scheduling Problem

The Urban Transit Network Design Problem (UTNDP) consist of two stages. The first stage is to create the physical route network (UTRP), whereas the second stage involves designing the schedules of the developed network (UTSP). This thesis has focused on the UTRP, creating effective urban transit routes. However, deciding when and how frequent a route should be serviced is also a vital part of optimizing a transit network. To increase the number of public transportation passengers, the schedules for the public vehicles must also be optimized. A good starting point could be to investigate the solution proposed by Nikolic and Teodorovic [2014], who solved both UTRP and UTSP using a BCO approach. Additional data must be provided to solve the UTSP efficiently. This data should include information about the demand between each bus stops different hours of the day.

Use the proposed system to optimize a transit network in a large city

The motivation for implementing the proposed system was initially to optimize the transit network in Trondheim. However, AtB (the bus service provider in Trondheim) does not possess the required data, such as the average demand values between each bus stop. An interesting approach would be to test the proposed system on a large transit network where the transit routes are manually designed, and where the service provider possesses the required data. These experiments would allow further investigation of the strength and weaknesses of the proposed system. It would also help determine how and if the system improves the current, manually designed, transit network. As stated above, the proposed system does not transfer well to large networks at this point, and changes must be done with the implementation before this is possible.

Bibliography

Alexanderson (2006). About the cover: Euler and kÖnigsberg's bridges: A historical view. *BULLETIN (New Series) OF THE AMERICAN MATHEMATICAL SOCIETY vol. 47*, page 567–573.

Alnaser, A. M., AlHeyasat, O., Abu-Ein, A. A.-K., Hatamleh, H. M. S., and Sharadqeh, A. A. M. (2012). Time comparing between java and c++ software. *Journal of Software Engineering and Applications*, pages 630–633.

ASA, T. (2015). Telenor. `http://www.telenor.no/`. Accessed: May 2015.

AtB (2015). Atb. `http://www.atb.no`. Accessed: January 2015.

Baaj and Mahmassani (1991). An ai-based approach for transit route system planning and design. *J. Adv. Transp. 25*, pages 187–209.

Baaj and Mahmassani (1995). Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research Part C 3*, pages 31–50.

Bagloee and Ceder (2011). Transit-network design methodology for actual-size networks. *Transportation Research Part B 45*, pages 1787–1804.

Beni and Wang (1989). Swarm intelligence. *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, page 425–428.

Bruggen, R. V. (2014). *Learning Neo4j*. Packt Publishing Ltd.

Chakroborty and Wivedi (2002). Optimal route network design for transit systems using genetic algorithms. *Eng. Optimiz. 34*, pages 83–100.

Chew and Lee (2012). A genetic algorithm for urban transit routing problem. *Int J. Mod. Phys.: Conf. Ser. 09*, pages 411–421.

Cisco (2015). Cisco. `http://www.cisco.com/`. Accessed: May 2015.

Cohen, P. R. and Howe, A. E. (1988). How evaluation guides ai research. *AI Magazine Volume 9 Number 4*, pages 35–43.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. Massachusetts Institute of Technology.

Dantzig and Ramser (1959). The truck dispatching problem. *Management Science*, pages 80–91.

DB-Engines (2015). Db-engines ranking of graph dbms. `http://db-engines.com/en/ranking/graph+dbms`. Accessed: March 2015.

Deneubourg, Aron, Goss, and Pasteels (1990). The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, pages 159–168.

Dias, Machado, Silva, and Abreu (2014). An inverted ant colony optimization approach to traffic. *Engineering Applications of Artificial Intelligence 36*, pages 122–133.

Dorigo and Gabardella (1997). Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transaction on Evolutionary Computation 1(1)*, pages 53–66.

Dorigo, Maniezzo, and Colorni (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems. Man and Cybernetics*, pages 1–13.

Dorigo, M., Birattari, M., and Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE COMPUTATIONAL INTELLIGENCE MAGAZINE*, pages 28–39.

Euler (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae 8*, pages 128–140.

Fan (2009). *Metaheuristic Methods for the Urban Transit Routing Problem.* PhD thesis, School of Computer Science Cardiff University.

Fan and Mumford (2010). A metaheuristic approach to the urban transit routing problem. *J. Heuristics 16*, pages 353–372.

Fan, Mumford, and Evans (2009). A simple multi-objective optimization algorithm for the urban transit routing problem. *IEEE Congress on Evolutionary Computation*, pages 1–7.

Ferrucci (2013). *Pro-active Dynamic Vehicle Routing.* Springer Berlin Heidelberg, Berlin, Heidelberg.

Google (2015). Google cloud platform: Machine types. `https://cloud.google.com/compute/docs/machine-types?hl=en_US&_ga=1.40008804.791529213.1430919997`. Accessed: May 2015.

Hsiao, Chuang, and Chien (2004). Ant colony optimization for best path planning. *International Symposium on Communication and Information Techniques.*

Jiang, Qingsong, and Huang (2010). An improved ant colony algorithm for urban transit network optimization. *Sixth International Conference on Natural Computation.*

Kechagiopoulos and Beligiannis (2014). Solving the urban transit routing problem using a particle swarm optimization based algorithm. *Applied Soft Computing*, pages 654–676.

Kidwai (1998). *Optimal Design of Bus Transit Network: A Genetic Algorithm Based Approach*. PhD thesis, Indian Institute of Technology.

Kitchenham, B. A. (2007). Guidelines for performing systematic literature reviews in software engineering. `https://www.cs.auckland.ac.nz/~norsaremah/2007%20Guidelines%20for%20performing%20SLR%20in%20SE%20v2.3.pdf`.

Kofod-Petersen (2014). How to do a structured literature review in computer science. `https://research.idi.ntnu.no/aimasters/files/SLR_HowTo.pdf`.

Lucic and Teodorovic (2003). Computing with bees: Attacking complex transportation engineering problems. *International Journal on Artificial Intelligence Tools*, pages 375–394.

Mandl (1980). Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, page 396–404.

Mandl, C. (1979). *Applied Network Optimization*. Academic Press, Stumpergasse 56, 1060 Wien, Austria.

Maven (2015). Apache maven. `https://maven.apache.org/`. Accessed: May 2015.

Miljøpakken (2014). Miljøpakkens mål. `http://miljopakken.no/om-miljoepakken/maal`. Accessed: November 2014.

Mumford, C. L. (2013). Supplementary material for: New heuristic and evolutionary operators for the multi-objective urban transit routing problem, cec 2013. `http://users.cs.cf.ac.uk/C.L.Mumford/Research%20Topics/UTRP/Outline.html`.

Neo Technology, I. (2015). Neo4j. `http://www.neo4j.com`. Accessed: February 2015.

Nikolic and Teodorovic (2014). A simultaneous transit network design and frequency settings: Computing with bees. *Expert Systems with Applications 41*, pages 7200–7209.

Poorzahedy and Safari (2011). An ant system application to the bus network design problem: an algorithm and a case study. *Int. J. Comput. Commun. Technol 2*, pages 105–110.

Robinson, Webber, and Eifrem (2013). *Graph Databases*. O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

Salehi-nezhad and Farrahi-Moghaddam (2007). An ant based algorithm approach to vehicle navigation. *First Joint Congress on Fuzzy and Intelligent Systems Ferdowsi University of Mashhad, Iran.*

Salehinejad and Talebi (2010). Dynamic fuzzy logic-ant colony system-based route selection system. *Applied Computational Intelligence and Soft Computing.*

Sedighpour, Ahmadi, Yousefikhoshbakht, Didehvar, and Rahmati (2014). Solving the open vehicle routing problem by a hybrid ant colony optimization. *Kuwait J. Sci. 41*, pages 139–162.

Sestoft, P. (2010). Numeric performance in c, c# and java. `https://www.itu.dk/people/sestoft/papers/numericperformance.pdf`.

Shi and Eberhart (1999). Empirical study of particle swarm optimization. *Evolutionary Computation vol. 3*, pages 1946–1950.

Shih and Mahmassani (1994). A design methodology for bus transit networks with coordinated operations. *Research report SWUTC/94/60016-1, Center of Transportation Research, University of Texas at Austin.*

Stutzle and Hoos (1999). Max-min ant system. *Future Generation Computer System*, pages 237–255.

Stützle, T. and Hoos, H. H. (2006). Max–min ant system. *Future Generation Computer Systems, vol. 16, no. 8*, pages 889–914.

Tripathi, Kuriger, and Wan (2009). An ant based simulation optimization for vehicle routing problem with stochastic demands. *Winter Simulation Conference.*

Walmart (2015). Walmart. `http://www.walmart.com/`. Accessed: May 2015.

Wan and Nolle (2009). Solving a multi-dimensional knapsack problem using a hybrid particle swarm optimization algorithm. *Conference: 23rd European Conference on Modelling and Simulation.*

Yang and Yu (2007). An parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering 22*, pages 44–55.

Zhang, Lu, and Fan (2010). The multi-objective optimization algorithm to a simple model of urban transit routing problem. *Sixth International Conference on Natural Computation*, pages 2812–2815.

Appendices

Appendix A

Structured Literature Review

A.1 Protocol

A.1.1 Defining the problem

To conduct a structured literature review it is vital to decide the problem, P, to be solved, and some constraints, C, to guide the search.

One of the goals for the environment package for transportation in Trondheim, "Greener Trondheim", is to improve the public transportation system [Miljøpakken, 2014]. From a meeting with AtB [AtB, 2015] we learned that the bus network in Trondheim never has been computational optimized, and the existing solution is purely based on experience. The problem formulation for this thesis was therefore based on the idea to improve todays solution by optimizing the bus routes using AI-methods.

- **P:** Optimizing the bus routes in Trondheim using AI-methods.
- **C:**
    1. To computationally optimize the bus routes in Trondheim we wanted to explore the possibility using methods from swarm intelligence. This idea came from an initial, non-structured literature review were we did a broad search among different artificial intelligence methods and route optimizing.

2. We believe that a part of solving the problem, P, is how we choose to represent the network of the bus routes in Trondheim. We have some experience with the graph database Neo4j. Neo4j has several benefits that we believe we can take advantage of when solving P, including a natural node-edge-structure and the possibility of saving information to both the nodes and edges. We envision that the nodes will represent bus stops, and the edges will represent the connectivity between the stops.

The problem and constraints is drawn to the following research question:

RQ 1: What is the state-of-the-art in solving vehicle routing problems using swarm intelligence methods and graph databases?

A.1.2 Search terms

Search terms were decided based on the defined research question, and formed into groups of synonyms:

- Group 1: Train, plane, bus, delivery
- Group 2: Path optimization, Scheduling optimization, Route optimization, Planning, Multimodal
- Group 3: Bee colony optimization, Particle swarm optimization, Swarm intelligence, Ant colony optimization, BCO, PSO, ACO
- Group 4: Transit, Transportation, Traffic, Vehicle
- Group 5: Artificial intelligence, AI, Machine learning
- Group 6: Multi-agent
- Group 7: Routing
- Group 8: Neo4j, Graph database

A.1.3 Complete search term

The search terms and the groups of synonyms were assembled into a complete search term:
(train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence"

OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi-agent" AND routing)

A.1.4 Inclusion criteria

The list of sources for retrieving relevant literature with the corresponding results is found in Section A.2 on page 94. To exclude irrelevant literature, some inclusion criteria were decided to ensure a level of relevance to the very first pool. First of all, duplicate literature, book of chapters, book of abstracts, book of references, literature not written in English, books, and literature with clearly irrelevant titles (for example literature from different research areas) were removed based on title. After that, we decided to filter out relevant literature based on the abstracts. Because we had relatively many sources to related literature after the initial filtering (367), we decided that we wanted the abstracts (or the keyword subsection) to explicitly mention swarm intelligence or algorithms associated with swarm intelligence, while it also described a problem connected to vehicle routing. For our literature review, we decided to use the inclusion criteria solely on the title, abstract, and keywords. After a discussion and reading a few abstracts we landed on the following inclusion criteria:

- The main concern is route optimization focusing on vehicles.
- The study focuses on the use of swarm intelligence.
- The literature must contain an abstract.
- The literature must still exist (some literature were removed from its original source).
- The literature must be free of charge.

After the inclusion criteria filtering, we had 42 sources to related literature, including scientific papers and master theses.

A.1.5 Quality criteria

The Quality Criteria was determined to ensure quality in the final papers and to filter away studies not theoretical relevant for our thesis, and are the following:

1. How relevant is it?
   (a) Is the problem of the research a vehicle routing problem?

(b) Is swarm intelligence the main optimization method?

2. Is there is a clear statement of the aim of the research?
3. Is the study put into a context of other studies and research?
4. Is a system of algorithmic design decisions justified?
5. Is the test data set reproducible?
6. Is the study algorithm reproducible?
7. Is the experimental procedure thoroughly explained and reproducible?
8. Is it clearly stated which algorithms their proposed algorithm is compared to?
9. Are the performance metrics used in the study explained and justified?
10. Are the test results thoroughly analyzed?
11. Does the test evidence support the findings presented?
12. Has the architecture been implemented (and published)?
13. Is the amount / quality of citation satisfactory? ($< \frac{1}{3}$ self-citation and > 10 citations)

A.1.5.1 Scoring

Point 1-13 was given a score, with the granularity of 0 (no), $\frac{1}{2}$ (partly), and 1 (yes). For this structured literature review, we wanted to emphasize on the quality criteria that covered the relevance. For retrieving the most relevant papers based on the content and not just the quality of the paper, Quality Criteria 1a on the previous page and 1b was multiplied by 3. Table A.1 on the next page shows the papers that were read and scored according to the quality criteria:

A.1.6 Selecting the final literature

When selecting the final literature, we decided to do this solely based on the quality criteria scores. The average score of the read literature was $12.95 \approx 13$. We decided that literature given a score of 1.5 above average were selected. After this sorting, we ended up with 11 final literature. These 11 literature are going to create the foundation of our thesis. Table A.2 shows the selected literature.

Title	1a	1b	2	3	4	5	6	7	8	9	10	11	12	13	Total
"A comprehensive review of firefly algorithms"	0	3	1	1	1	0	1	0	1	0.5	0	0	0	1	9
"Adaptive Comprehensive Learning Bacterial Foraging Optimization and Its Application on Vehicle Routing Problem with Time Windows"	1.5	1.5	0.5	1	0.5	1	1	1	1	1	1	0	0	1	12
"Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model"	1.5	1.5	1	1	1	0.5	0.5	1	0	1	1	1	0	1	12
"A Design of Intelligent and Autonomous Public Transportation System by Co-evolution"	3	1.5	0	0	0.5	0	0	0	0	0	0	0	0	0.5	5.5
"A fast solution method for the time-dependent orienteering problem"	3	1.5	1	1	0.5	1	1	1	1	1	1	1	0	1	15
"Agent-based Simulation for UAV Swarm Mission Planning and Execution"	0	1.5	1	1	0.5	0.5	1	0.5	1	0.5	0.5	0.5	0	1	9.5
"A hybrid particle swarm optimization approach for the sequential ordering problem"	1.5	3	1	1	1	1	1	1	1	1	0.5	0.5	0	0.5	14
"An ant based algorithm approach to vehicle navigation"	3	3	1	0.5	1	0.5	1	1	0.5	1	1	0.5	0	0.5	14.5
"An Ant Based Simulation Optimization for Vehicle Routing Problem with Stochastic Demands"	1.5	3	1	1	1	1	1	1	1	0.5	1	0.5	0	1	14.5
"An Ant Colony Optimization Approach to Solve Cooperative Transportation Planning Problems"	1.5	1.5	0	1	0.5	0.5	1	1	1	1	1	1	0	1	12
"An Ant System application to the Bus Network Design Problem: an algorithm and a case study"	3	3	1	1	1	1	1	1	1	1	1	1	0	1	17
"An exploration of the literature on the use of 'swarm intelligence-based techniques' for public service problems"	1.5	3	1	0.5	1	1	0.5	1	0.5	1	0.5	0.5	0	1	13
textit"An improved Ant Colony algorithm for Urban Transit Network Optimization"	3	3	1	1	1	1	1	1	1	1	1	1	0	1	17
"An Inverted Ant Colony Optimization approach to traffic"	1.5	3	1	1	1	1	1	1	0.5	1	1	0.5	0	1	14.5
"Ant Colony Optimization"	1.5	3	0.5	0.5	0.5	1	0.5	0.5	0.5	0.5	0	0	0	0	9
"Ant colony optimization for best path planning"	3	3	1	1	1	1	1	1	0.5	1	1	1	0	0.5	16
"Ant colony optimization techniques for the vehicle routing problem"	3	3	1	0	0.5	0.5	0.5	0.5	1	1	1	1	0	1	14
"Ant dispersion routing for traffic optimization"	1.5	1.5	1	1	1	1	1	1	1	1	1	0.5	0	1	13.5
"A parallel ant colony algorithm for bus network optimization"	3	3	1	1	0.5	1	1	1	0.5	1	1	1	0	1	16
"A review of ant algorithms"	1.5	3	0.5	1	0.5	1	1	1	1	1	1	0.5	0	1	14
"A simultaneous transit network design and frequency setting: Computing with bees"	3	3	1	1	1	1	1	1	1	1	1	1	0	1	17
"A Study on Bus Routing Problem: An Ant Colony Optimization Algorithm Approach"	3	3	1	1	1	0.5	0.5	0.5	1	0.5	0.5	0.5	0	0	13
"A Swarm Based Method for Solving Transit Network Design Problem"	3	3	0.5	1	0.5	0.5	1	1	1	1	0.5	0.5	0	0.5	14
"Bi-objective bimodal urban road network design using hybrid metaheuristics"	1.5	1.5	1	1	0.5	1	1	1	1	1	1	1	0	1	13.5
"Bus Transit Service Optimization–The State-of-the-Art. State-of-the-Practice, and Challenges"	1.5	0	1	0	1	1	0.5	0.5	0.5	0.5	0.5	0.5	0	0.5	8
"Combining new Fast Opposite Gradient Search with Ant Colony Optimization for solving travelling salesman problem"	0	3	1	1	0.5	1	1	1	1	1	1	0.5	0	1	13
"Computing with bees: Attacking complex transportation engineering problems"	1.5	3	1	1	0.5	0.5	0.5	1	0	0.5	1	0.5	0.5	1	12.5
"Data mining with various optimization methods"	0	1.5	1	1	0.5	0	1	0.5	1	1	1	1	0.5	1	11
"Designing a multimodal feeder network by covering stops with different modes"	1.5	3	1	1	1	0	0.5	1	0	1	1	1	0.5	1	13.5
"Dynamic Fuzzy Logic-Ant Colony System-Based Route Selection System"	3	3	0.5	1	1	0.5	0.5	0.5	1	1	1	1	0.5	1	15.5
"Multimodal Feeder Network Design Problem: Ant Colony Optimization Approach"	1.5	3	1	1	0.5	0	1	0.5	0	1	1	1	0.5	1	13
"Optimization of a Transit Services Model with a Feeder Bus and Rail System Using Metaheuristic Algorithms"	3	1.5	0.5	1	1	0	1	0.5	1	1	1	1	0.5	1	14
"Optimization of Large Transport Networks Using Ant Colony Heuristic"	0	3	0.5	0.5	0.5	0.5	1	0.5	0	1	0.5	1	0.5	1	10.5
"Optimizing bus transit network with parallel ant colony algorithm"	1.5	3	1	0.5	0.5	1	1	0	1	1	1	1	0.5	1	14
"Real-time route planning of the public transportation system"	3	3	1	0	0.5	0.5	1	0.5	0.5	1	0.5	1	0.5	1	14
"Route Optimization for Bus Dispatching Based on Improved Ant Colony Algorithm"	3	3	0.5	0	0.5	0.5	1	0.5	0	0	0.5	1	0.5	0.5	11.5
"Solving the open vehicle routing problem by a hybrid ant colony optimization"	3	3	1	1	1	1	1	1	1	1	1	1	0.5	1	17.5
"Solving the Urban Transit Routing Problem using a particle swarm optimization based algorithm"	3	3	1	1	1	1	1	1	1	1	1	1	0.5	1	17.5
"The Application of Artificial Intelligence Hybrid in Traffic Flow"	0	3	0.5	0	0	0	0	0	0	0	0	0	0	1	4.5
"Transit network design by Bee Colony Optimization"	1.5	3	1	1	1	1	1	1	0.5	0	0.5	1	0.5	1	14
"Transportation Modeling: An Artificial Life Approach"	1.5	3	1	0.5	0.5	0	0.5	0.5	0.5	0.5	0.5	1	0.5	1	11.5
"Transport Modeling by Multi-Agent Systems: A Swarm Intelligence Approach"	3	3	0.5	0.5	0.5	0	0	0	0.5	0	0	0	0	1	9

Table A.1: Quality criteria scoring

Title	Author
"An ant based algorithm approach to vehicle navigation"	Salehi-nezhad and Farrahi-Moghaddam
"An Ant Based Simulation Optimization for Vehicle Routing Problem with Stochastic Demands"	Tripathi et al.
"An Ant System application to the Bus Network Design Problem: an algorithm and a case study "	Poorzahedy and Safari
"An improved Ant Colony algorithm for Urban Transit Network Optimization"	Jiang et al.
"An Inverted Ant Colony Optimization approach to traffic"	Dias et al.
"Ant colony optimization for best path planning"	Hsiao et al.
"A parallel ant colony algorithm for bus network optimization"	Yang et al.
"A simultaneous transit network design and frequency setting: Computing with bees"	Nikolić and Teodorović
"Dynamic Fuzzy Logic-Ant Colony System-Based Route Selection System"	Salehinejad and Talebi
"Solving the open vehicle routing problem by a hybrid ant colony optimization"	Sedighpour et al.
"Solving the Urban Transit Routing Problem using a particle swarm optimization based algorithm"	Kechagiopoulos and Beligiannis

Table A.2: Final selected literature

A.2 Search engines and search strings

In this structured literature review, we decided to search in seven different search engines. The process of deciding which search engines to use was strongly influenced by the suggestions in Kofod-Petersen [2014]. The complete search term, described in Section A.1.3 on page 90, is built on terms from seven different groups. In addition to the search of the complete search, a search consisting of one additional group (group 8) was conducted in each of the different search engines. Group 8 consists of the words "neo4j" and "graph database". This additional search was done to investigate if the combination of swarm technology and graph databases to solve a route optimization problem already had been studied. The results from our search show that our search term combined with "neo4j" or "graph database"

gave zero findings.

A.2.1 ACM Digital Library

Notes: ACM Digital Library did not support a mix of ANDs and ORs in its initial input field, but this was possible in advanced search. The search string was not modified, and the first search gave a satisfactory number of results.

Queries:

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing

Date of search: 2014-11-10

Results: 19

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing AND ("graph database" OR neo4j)

Date of search: 2014-11-10

Results: 0

A.2.2 ScienceDirect

Notes: In ScienceDirects advanced search it was only possible to perform a full text search. The first search was within "all sciences" and this retrieved 100 results. The next search was therefore just within "Computer Science", which gave less, but a lot more relevant literature. In addition to this books were excluded from the results.

Queries:

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing

Date of search: 2014-11-10

Results: 60

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing AND ("graph database" OR neo4j)

Date of search: 2014-11-10

Results: 0

A.2.3 CiteSeer

Notes: In CiteSeer you cannot perform a search within title, abstract and keywords at the same time. It was therefore conducted a full text search by adding the element text:() to the query. Some of the retrieved literature had an unknown title with unknown authors, so theese were excluded from the results.

Queries:

- text:((Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing)

Date of search: 2014-11-10

Results: 27

- text:((Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing) AND ("graph database" OR neo4j)

Date of search: 2014-11-10

Results: 0

A.2.4 SpringerLink

Notes: In SpringerLinks advanced search it was only possible to find literature with either all the words, the exact phrase or at least one of the words in the search string. For this reason the whole boolean search string was used in the initial input field. The first search gave 200 results, so the next search was only conducted within "computer science" and "engineering". In addition to this only results within "articles" were selected.

Queries:

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing

Date of search: 2014-11-10

Results: 28

- (Train OR plane OR bus OR delivery) AND ("path optimization" OR "scheduling optimization" OR "route optimization" OR planning OR multimodal) AND ("bee colony optimization" OR "particle swarm optimization" OR

"swarm intelligence" OR "ant colony optimization" OR bco OR pso OR aco) AND (transit OR transportation OR traffic OR vehicle) AND ("artificial intelligence" OR ai OR "machine learning") AND "multi agent" AND routing AND ("graph database" OR neo4j)

Date of search: 2014-11-10

Results: 0

A.2.5 IEEE Xplore

Notes: The search is done in full text, including metadata. The search string had to be changed to fulfill IEEE's criteria that the string only should contain 15 search terms.

Queries:

- ("public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization")

Results: 45

Date of search: 2014-11-10

- ("public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization") AND neo4j)

Results: 0

Date of search: 2014-11-10

- ("public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization") AND "graph database")

Results: 0

Date of search: 2014-11-10

A.2.6 ISI Web of Knowledge

Notes: In Web of Knowledge you cannot perform at full text search, and must choose to search in "Topic", "Title", "Author", "Author Identifiers", "Editor", "Group Author", "Publication Name", "DOI" or "Year Published". We decided to use "Topic", "Title" and "Publication Name" because it seemed the most relevant to our search terms. The search was done in "All databases", but only in the "COMPUTER SCIENCE" research area. The original search string (see table **??**) had to be modified, because it gave no results in Web Of Knowledge. A few terms were therefor excluded, and a few AND's were switched with OR's.

Queries:

- ("public transportation" OR traffic OR transportation OR transit OR "scheduling optimization" OR "path optimization" OR "route optimization" OR planning OR multimodal OR routing) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR pso OR aco OR bco) AND ("artificial intelligence" OR ai OR "machine learning")

Results: 47 (Topic) + 0 (Title) + 0 (Publication Name)

Date of search: 2014-11-11

- ("public transportation" OR traffic OR transportation OR transit OR "scheduling optimization" OR "path optimization" OR "route optimization" OR planning OR multimodal OR routing) AND ("bee colony optimization" OR "particle swarm optimization" OR "swarm intelligence" OR "ant colony optimization" OR pso OR aco OR bco) AND ("artificial intelligence" OR ai OR "machine learning") AND (neo4j OR "graph database")

Results: 0 (Topic) + 0 (Title) + 0 (Publication Name)

Date of search: 2014-11-11

A.2.7 Google Scholar

Notes: Google Scholar only allows very short search strings, and we were therefor forced to split the query into smaller pieces and do mulitple search, for so to add the results togheter. The original search string had to be modified for making the splitting tolerable and effective.

Queries:

- "public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND "bee colony optimization"

Results: 21

Date of search: 2014-11-10

- "public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND "bee colony optimization" AND neo4j

Results: 0

Date of search: 2014-11-10

- "public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal) AND (transit OR traffic) AND ("artificial intelligence" OR ai OR "machine learning") AND routing AND "bee colony optimization" AND "graph database"

Results: 0

Date of search: 2014-11-10

- "public transportation" AND ("path optimization" OR "route optimization" OR planning OR multimodal)AND(transit OR traffic)AND("artificial intelligence" OR ai OR "machine learning")AND routing AND "particle swarm optimization"

Results: 78

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal)AND(transit OR traffic)AND(“artificial intelligence” OR ai OR “machine learning”)AND routing AND “particle swarm optimization” AND neo4j

Results: 0

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal)AND(transit OR traffic)AND(“artificial intelligence” OR ai OR “machine learning”)AND routing AND “particle swarm optimization” AND “graph database”

Results: 0

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “swarm intelligence”

Results: 76

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “swarm intelligence” AND neo4j

Results: 0

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “swarm intelligence” AND “graph database”

Results: 0

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “ant colony optimization”

Results: 119

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “ant colony optimization” AND neo4j

Results: 0

Date of search: 2014-11-10

- “public transportation” AND (“path optimization” OR “route optimization” OR planning OR multimodal) AND (transit OR traffic) AND (“artificial intelligence” OR ai OR “machine learning”) AND routing AND “ant colony optimization” AND “graph database”

Results: 0

Date of search: 2014-11-10

Appendix B

Input Data

B.1 Mandl's Network

A method is, as mentioned, designed and implemented to produce a realistic transit network based on the data from Mandl [Mandl, 1979]. The data includes a network of 15 nodes and 21 edges, in addition to the travel times and travel demand for each edge.

The data set is downloaded from Mumford [2013], and is presented in Table B.1 on the next page, Table B.1 on page 105, and Table B.3 on page 105.

The file presented in Table B.1 on the following page, includes 16 lines, with a number indicating the amount of nodes, and the (x,y) coordinates for the 15 nodes. These coordinates was not supplied in Mandl's literature, so they are retrieved from Mumford [2013] and are approximate for the picture to be drawn.

	x	y
1	1	9
2	3	8
3	4.5	7.75
4	2.75	6.2
5	0.8	6.6
6	4.6	6
7	7	4.5
8	5.5	5
9	8.5	6.8
10	5.8	2.25
11	3.8	2.25
12	1.3	3.5
13	5.25	1
14	6.7	1.75
15	6.75	5.8

Table B.1: Coordinates for Mandl's Network

Table B.2 shows the content of the travel time file. The presented travel times matrix gives the travel times it takes in minutes between the nodes. This matrix is symmetrical, travel times between each node and itself are zero, and "Inf" indicates that there is no direct link between the nodes.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	8	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
2	8	0	2	3	6	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
3	Inf	2	0	Inf	Inf	3	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
4	Inf	3	Inf	0	4	4	Inf	Inf	Inf	Inf	Inf	10	Inf	Inf	Inf
5	Inf	6	Inf	4	0	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
6	Inf	Inf	3	4	Inf	0	Inf	2	Inf	Inf	Inf	Inf	Inf	Inf	3
7	Inf	Inf	Inf	Inf	Inf	Inf	0	Inf	Inf	7	Inf	Inf	Inf	Inf	2
8	Inf	Inf	Inf	Inf	Inf	2	Inf	0	Inf	8	Inf	Inf	Inf	Inf	2
9	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	0	Inf	Inf	Inf	Inf	Inf	8
10	Inf	Inf	Inf	Inf	Inf	Inf	7	8	Inf	0	5	Inf	10	8	Inf
11	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	5	0	10	5	Inf	Inf
12	Inf	Inf	Inf	10	Inf	Inf	Inf	Inf	Inf	Inf	10	0	Inf	Inf	Inf
13	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	10	5	Inf	0	2	Inf
14	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	8	Inf	Inf	2	0	Inf
15	Inf	Inf	Inf	Inf	Inf	3	2	2	8	Inf	Inf	Inf	Inf	Inf	0

Table B.2: Travel times for Mandl's Network

Table B.3 on the facing page presents the demand file. The demand matrix in this file shows the travel demand between each node pair, which is the average number of passenger trips per day. This matrix is also symmetrical. There is no demand

either to or from node 15, but it there is demand to node 9, which is only connected to node 15. For coding reasons, is the numbers in the main diagonal (from to left corner to bottom right corner) changed from all zero to all 'a's.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	a	400	200	60	80	150	75	75	30	160	30	25	35	0	0
2	400	a	50	120	20	180	90	90	15	130	20	10	10	5	0
3	200	50	a	40	60	180	90	90	15	45	20	10	10	5	0
4	60	120	40	a	50	100	50	50	15	240	40	25	10	5	0
5	80	20	60	50	a	50	25	25	10	120	20	15	5	0	0
6	150	180	180	100	50	a	100	100	30	880	60	15	15	10	0
7	75	90	90	50	25	100	a	50	15	440	35	10	10	5	0
8	75	90	90	50	25	100	50	a	15	440	35	10	10	5	0
9	30	15	15	15	10	30	15	15	a	140	20	5	0	0	0
10	160	130	45	240	120	880	440	440	140	a	600	250	500	200	0
11	30	20	20	40	20	60	35	35	20	600	a	75	95	15	0
12	25	10	10	25	15	15	10	10	5	250	75	a	70	0	0
13	35	10	10	10	5	15	10	10	0	500	95	70	a	45	0
14	0	5	5	5	0	10	5	5	0	200	15	0	45	a	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a

Table B.3: Demand for Mandl's Network

Appendix C

Experimental Results

C.1 Parameter settings results

Parameter	s	i	E	CI^1	STD^2	BEST	WORST
s	**10**[3]	50	10%	117.244[1] ± 3.854	10.700	88.364	137.997
	50	50	10%	105.424 ± 2.369	6.619	90.076	114.843
	100	50	10%	99.890 ± 2.193	6.128	87.178	108.688
	125	50	10%	**97.994 ± 2.267**	**6.334**	**80.484**	**105.908**
i	50	**10**	10%	113.209 ± 3.205	8.957	94.551	130.521
	50	**50**	10%	105.044 ± 2.667	7.454	90.293	120.946
	50	**100**	10%	101.013 ± 1.802	4.952	82.183	109.379
	50	**125**	10%	**96.144 ± 2.106**	**5.786**	**82.709**	**107.362**
E	50	50	**10%**	104.395 ± 2.125	5.938	92.757	115.734
	50	50	**25%**	**103.274 ± 2.998**	**8.379**	**86.917**	**120.811**
	50	50	**50%**	103.486 ± 2.576	7.200	83.245	116.951
	50	50	**90%**	104.252 ± 2.667	7.452	85.033	116.634

Table C.1: Steps with the corresponding results from the parameter settings experiment (parameter s, i and E)

[1]: Confidence Interval of Total Fitness (confidence level: 95%)
[2]: Population Standard Deviation
[3]: On average 14.467% of the iterations of each run did not create any ants that satisfied the initial Constraint 4 described in Section 4.3.1 on page 42.

Parameter	CA	AF	p_b	CI^1	STD^2	BEST	WORST
CA	**0%**	10%	0.0	105.66 ± 1.917	6.915	89.463	118.669
	5%	10%	0.0	104.581 ± 1.576	5.686	93.975	119.024
	10%	10%	0.0	104.064 ± 1.882	6.791	90.058	118.076
	25%	10%	0.0	**103.597 ± 2.082**	**7.511**	**88.545**	**119.022**
	50%	10%	0.0	105.100 ± 1.726	6.228	86.434	118.981
	100%	10%	0.0	110.135 ± 2.26	8.153	91.194	125.927
AF	10%	**0%**	0.0	105.747 ± 1.487	5.364	89.031	114.771
	10%	**5%**	0.0	**104.389 ± 1.733**	**6.252**	**92.358**	**118.361**
	10%	**10%**	0.0	104.739 ± 1.861	6.714	91.345	123.648
	10%	**25%**	0.0	104.632 ± 1.747	6.302	87.838	118.012
	10%	**50%**	0.0	106.327 ± 2.214	7.986	89.808	119.931
	10%	**100%**	0.0	120.245 ± 3.546	12.792	94.116	145.600
p_b	10%	5%	**0.0**	104.882 ± 2.408	6.728	78.721	113.019
	10%	5%	**0.1**	104.101 ± 2.303	6.435	89.282	116.23
	10%	5%	**0.5**	103.192 ± 2.837	7.928	81.306	115.088
	10%	5%	**0.9**	103.058 ± 1.665	4.653	92.982	111.865
	10%	5%	**1.3**	**102.579 ± 2.558**	**7.147**	**85.033**	**115.895**

Table C.2: Steps with the corresponding results from the parameter settings experiments (parameter CA, AF and p_b)

[1]: Confidence Interval of Total Fitness (confidence level: 95%)
[2]: Population Standard Deviation

C.2 Performance comparison results

Algorithm	$d_0(\%)$	$d_1(\%)$	$d_2(\%)$	$d_{unsat}(\%)$	ATT
ACO Best	84.07	15.03	0.90	0.00	10.08
ACO Average	81.92	16.13	1.86	0.09	10.43
ACO Median	82.66	15.29	1.16	0.00	10.42
ACO Worst	64.93	21.19	11.5	2.38	10.79
ACO Standard Deviation	4.87	3.75	1.83	0.37	0.32
CSS Best	87.73	10.98	1.28	0.00	10.03
CSS Average	85.21	13.49	1.30	0.00	10.27
CSS Median	85.81	13.29	1.09	0.00	10.26
CSS Worst	76.56	22.16	1.28	0.00	10.01
CSS Standard Deviation	2.66	2.70	0.84	-	0.18

Table C.3: Comparing the route sets of ACO and CSS, having four routes (all results)