Disruption Management at NS

Crew scheduling using Multi Agent System techniques

F. Vermeulen
1660179
August 2011

Master Thesis

Artificial Intelligence
Department of Artificial Intelligence
University of Groningen, The Netherlands

Internal supervisor:
Dr. M.A. Wiering (Artificial Intelligence, University of Groningen)

External supervisor:
M.Sc. S. Dalmolen (Groningen)

rijksuniversiteit groningen / faculteit wiskunde en natuurwetenschappen / kunstmatige intelligentie

Abstract

Disruption management is an important topic in railway management and control. When a disruption occurs the original operational plan may become infeasible and will need to be revised to make it suitable for the new environment. The Dutch Railways (NS) address this problem and stress the importance of getting back to stable operations as efficiently as possible, especially in case of major incidents and disruptions. This research deals with a specific part of the problem, namely the planning of personnel on a newly generated train timetable. The current study employs Multi-Agent System (MAS) techniques, specifically Multiple Ant Colony Systems (MACS), to solve the scheduling problem. This research shows that MACS in itself is not enough to solve the scheduling problem, and heuristics are necessary to reach solutions of acceptable quality, which leads to the MACS$^+$ algorithm. The experimental results obtained with the MACS$^+$ algorithm were generated with the aid of a dataset supplied by the NS and subsequently compared to their best solution. Moreover, the results were also compared to a purely random choosing of tasks and a greedy algorithm. The quality of the NS solution was not equalled, but our system performed far better than random and greedy algorithms.

Contents

Chapter 1

Introduction

Disruption management is an important topic in railway management and control. When a disruption occurs the original operational plan may become infeasible and will need to be revised to make it suitable for the new environment. The Dutch Railways (NS) address this problem and stress the importance of getting back to stable operations in as little time and with the lowest costs as possible, especially in case of major incidents and disruptions. This boils down to especially good new (or adjusted) plans for both personnel and trains. Most important are good alternative plans for personnel (the train drivers and conductors). Such a plan is optimized to reduce the amount of extra (over)work hours, the amount of taxi kilometres, and even hotel costs due to the fact that personnel does not make it home in time any more. Repositioning of trains is perceived less of an issue, as trains are relatively easily interchangeable (NS utilizes only a handful of different train types, and train types are generally assigned to specific lines – a major disruption such as a broken wire, therefore tends to result in a certain amount of trains on the one side, and a certain amount at the other side). Personnel however strongly prefers to make it home at the end of their working day; preferably more-or-less according to the time planned for in the original scheme.

There are on average 3 disruptions occurring per day per route on the Dutch railway system. A route is the path a train travels, for example Groningen to Schiphol Airport. Delays are even more frequent: on average 450 trains experience one or more delays ($>$ 3 minutes) per day (out of more than 5000 daily trains) [4]. To be able to react adequately to a disruption, large or small, a re-scheduling system has to be flexible to be able to deal with all disruptions and fast to be able to generate a solution in near real-time. Thus, information from different sources needs to be acquired. Not only the static information containing the current NS timetable is needed, but also information about the current location of personnel and their capabilities as well as the current location of trains. Moreover, information about the expected duration of the disruption is very useful. For planned maintenance the duration of the disruption can be estimated with relatively high accuracy, whereas the expected duration of a malfunction or accident could be far more difficult to estimate. When a disruption is resolved the normal timetable should be resumed as soon as possible with a smooth transition from the temporary schedule to the original.

1.1 Problem description

Immediately following a disruption a new crew schedule has to be created that works around the problem that has occurred. This crew schedule is based on the supplied rolling stock schedule and the current location of the available crew. The deviation from the original schedule before the disruption has to be kept to a minimum to ensure a smooth transition back to the original schedule after the disruption has been resolved. Moreover, the new schedule has to be as efficient as possible with respect to constraints such as minimal idle time of employees and a lunch break at an appropriate interval during a shift.

It is imperative that a new schedule is formed as soon as possible after a disruption. Creating a schedule in near real-time not only greatly improves the ability of personnel to perform their jobs, it also improves the information that can be supplied to travellers, although the latter is not the primary goal of this research. Furthermore, the faster a new timetable can be constructed, the faster passengers will be able to continue with their journey.

The current study employs Multi-Agent System (MAS) techniques, specifically Multiple Ant Colony Systems (MACS) [13], to tackle the scheduling problem. As the analogy between multiple employees working together to service trains according to a timetable and multiple agents working together to solve a problem is easily made, we assume that MAS techniques lend themselves well for a crew re-scheduling problem. MACS was developed to solve the Bus Allocation Problem (BAP), where n bus lines need to be constructed between m bus stops with the aim to minimize the Average Travel Time (ATT) of passengers travelling from one bus stop to another. Here ants from multiple ant colonies work together in an effort to find an optimal bus line schedule. In the crew scheduling situation instead of vehicles, crew members are scheduled which have different constraints, for example a maximum number of consecutive working hours and the right to take a lunch break if a shift exceeds a certain length. Therefore, the basic MACS structure will need to be extended to make it suitable for the more complex problem of crew re-scheduling.

It is important to note that this research will focus on the scheduling of crew on a given timetable for the rolling stock. Thus, here it is assumed that immediately following a disruption a new rolling stock schedule has been created that will form the basis on which the crew scheduling system will create a new crew timetable for the train drivers and conductors.

1.2 Research questions

As mentioned in the problem description above (section 1.1), we will focus on the scheduling of personnel on a predefined new time table. The first problem we will aim to solve is that of scheduling personnel for an entire work day from one base station with a supplied time table for the rolling stock. Which leads us to the main research question this research will focus on:

How well does the MACS algorithm lend itself for a crew scheduling problem in a railway time table setting?

The results obtained from answering this main research question will give insight in the capabilities of the crew scheduling system. With these insights we will then move on to the second research question:

How well does the MACS algorithm lend itself for a crew re-scheduling problem in a railway disruption management setting?

The difficulty of re-scheduling does not only lie in creating a feasible schedule, but more importantly in creating the schedule fast enough for it to be employed virtually immediately after a disruption has occurred. Apart from the speed with which a new schedule needs to be created there is the added constraint of making sure that the schedule will smoothly transfer to the original situation when the disruption is resolved and normal operations resume. Moreover, when a disruption occurs employees are situated in various locations from where they will need to be included in the new crew schedule.

The crew scheduling algorithm will be tested on a dataset supplied by NSR Logistics Innovation which contains real world data for one working day with the Enschede train station as a base (section 3.1). NSR Logistics Innovation is the research division of the Dutch Railways and is responsible for creating optimal rolling stock and crew schedules, as well as researching the best way to handle disruptions.

Experimental results will be compared with a given solution obtained by the current best algorithm developed by NSR Logistics Innovation. Prototype Alpha will contain a basic set of constraints with which experiments will be conducted. The results from these experiments will form the basis for Prototype Beta that will contain the full set of constraints. Experimental results from Prototype Beta will be compared to the solution supplied by NSR Logistics Innovation.

The remainder of this thesis is organized as follows; in the next chapter the theoretical background is given, which includes early work and more recent approaches in various areas of public transport scheduling as well as approaches utilizing Ant Systems. In chapter 3 the methodology and implementation of our system is given. The MACS system is explained in detail as well as the extensions made, resulting in MACS$^+$, which can be used for crew scheduling. Next the implementation specifics, experiments and results of Prototype Alpha will be discussed in detail in chapter 4. In chapter 5 Prototype Beta is discussed. Finally, chapter 6 contains a discussion and conclusions as well as a section devoted to future work.

Chapter 2

Theoretical background

2.1 Scheduling

The crew scheduling problem is part of the combinatorial optimization problems and can be seen as a variation on the Travelling Salesman Problem (TSP) where the shortest path needs to be found while visiting n cities exactly once. In a crew scheduling problem the cities from the TSP are replaced by tasks that need to be completed and a lowest cost path through the tasks needs to be found to create one shift. In contrast with a TSP there are multiple crew members that need to create shifts. Thus, multiple paths then form a crew schedule for one work day. As the combinatorial optimization problem of crew scheduling is part of the NP-hard problems, finding a solution becomes very complex when considering a dataset containing 11000 tasks (the NS Enschede dataset in the current research is discussed in section 3.1). First early work on scheduling is discussed after which we move on to more recent approaches and examples of Ant Colony Optimization used in crew scheduling.

2.1.1 Early work

Crew scheduling automation in a public transport setting is nothing new. Wren and Rousseau [38] mention that there has been early research on the subject where the problem has been formulated in linear programming models. However, the hardware technology available before the late 1970s only allowed for very small models to be solved, thus it remained unusable for large scale real-world environments. This meant that research mainly focused on the development of heuristics in an effort to reduce the computation time needed to solve the models. However, by the beginning of the 1980's it was recognized that heuristics alone were insufficient for general use without first conducting considerable research on the specific problem at hand. Because of this, subsequent research moved towards combining both heuristics with mathematical programming methods in order to develop more general systems that could be applied in many different operating environments.

Although subsequent systems were able to help with the creation of a new schedule, still a lot of manual scheduling by scheduling experts editing the created schedules was needed where no solution was found. Only in the last few years there have been systems that can create schedules faster, and more importantly, more efficiently than manual schedulers.

2.1.2 Recent approaches

There has been research on the subject for either bus, airline, and train industries [38, 8, 7], including mathematical solutions [6] and proposals for the use of Multi-Agent Systems [34]. Although yielding good results, the mathematical approaches are slow, taking up to several hours to create a schedule. An example is TRACSII [24], which uses a set-covering model to select a feasible schedule from a previously generated set of shifts. The time needed to generate a schedule with such a mathematical approach makes it unsuitable for re-scheduling purposes, where minimizing computation time is of great importance.

In trying to solve the scheduling problem, some approaches included the planning of rolling stock as well as personnel within one solution. This means being able to simultaneously plan trains (or buses, airplanes) and the crews needed to operate them. This was done in an effort to come to a better solution to the two problems as opposed to solving them separately [36]. Moreover, Freling et al. [26] mention that because vehicles (buses in this case) are often more flexible to schedule than crews, it may be inefficient to schedule vehicles without considering crew scheduling. They state that vehicle oriented characteristics of crew scheduling may affect the extent to which the integration of vehicle and crew scheduling is beneficial compared with the traditional sequential approach of first creating a vehicle schedule before building a timetable for the crew. Examples of these characteristics are; a restricted number of changeovers, restricted deadheading, extra start-up time on a new vehicle, compulsory continuous attendance when a vehicle is waiting, among others. However, it lies outside the scope of the current study to consider the simultaneous scheduling of rolling stock and crew. Moreover, the Dutch Railways (NS) have, as many other railway companies in Europe do, a cyclic schedule, which makes it easy to remember for passengers but has the disadvantage that it can not be tuned to perfectly align with a crew schedule to obtain an optimal solution.

Another problem that has emerged while trying to solve crew scheduling problems for the NS is the satisfaction of personnel. When creating a schedule for rolling stock there are no problems concerning lunch breaks, maximum number of consecutive working hours, repetition of the same route, and union laws. However, when creating a crew schedule these constraints have to be taken very seriously, otherwise a newly formed schedule may lead to strikes [2]. Although there are more constraints, the meal break seems to be a very hard problem to tackle. In most crew scheduling situations efficient chains of meal breaks need to be formed. For example, in a bus driver scheduling problem, an early driver who works a partial shift, is relieved by a driver who has taken a bus back to the depot, and himself relieves another driver after taking a meal [38].

Abbink et al. [2] used a set-covering model consisting of a duty-generation module and a duty-selection module. This algorithm starts by generating large amounts of possible duties that satisfy strict rules pertaining to crew demands. Subsequently, the duty-selection module selects a subset of these millions of duties and aims to find the subset that covers all the trips while minimizing the total costs. A main disadvantage of this process is the several hours of computation time needed to create a schedule, which is irrelevant when scheduling for the coming month or year, but computation time becomes an issue when re-scheduling is needed when an unforeseen disruption occurs.

2.2 Re-scheduling

Up until recently most crew scheduling research has focused on creating the schedule from scratch. This has the advantage of not having to take into account previous schedules, which allows more freedom in the scheduling phase. However, the Crew Re-Scheduling Problem (CRSP) differs from this, because there is a previous schedule that has to be taken into account. The quality of the new schedule depends, among other criteria, on the deviation from the original. Keeping changes to a minimum is preferred [30]. Moreover, once the disruption that triggered the creation of a new schedule has been resolved, normal operations should not only be resumed as fast a possible, but the transition from the temporary schedule to the original should be as smooth as possible.

Huisman [30] focuses on re-scheduling for planned disruptions (i.e. maintenance), which means that a new schedule can be constructed beforehand without the need for short computation times. Apart from the planned disruptions there are also unforeseen incidents that can cause a disruption to the schedule. The new schedule needs to be constructed in near real-time, therefore a comparatively slow system is not sufficient when tackling such a disruption.

There are two general methods for reducing the problem space, and thus the computation time, in optimization problems pertaining to re-scheduling mentioned in [11]. First, the *time window technique* is applied, which aims to reduce the search horizon by limiting the time window from the time the disruption occurred up to a limited amount of time into the future. The second step limits the amount of crew members included into the solution process, which will decrease computation time, but might also decrease the quality of the solution. The scheduling system in this study will include a limited search horizon to reduce computation time.

2.2.1 Agent based re-scheduling

In Abbink et al. [4] the suitability of a decentralized, actor-agent based approach to crew re-scheduling is explored in order to establish if a Multi-Agent System (MAS) is sufficiently capable of being put to work in a real-world crew re-scheduling system. When a disruption occurs, a driver-agent that is directly affected by this becomes a so called *team leader* and starts a *task exchange* process with other agents. It could occur that another agent can take over a task without additional costs (e.g. a deadheading trip becomes a task), but the takeover could also result in extra conflicts that subsequently have to be resolved. This process is considered complete if all conflicts are resolved, or if they are moved a sufficient amount of time into the future to be resolved at a later point in time.

Further research on agent-based re-scheduling (in the bus public transportation area) has been done by Shibghatullah et al. [34], where it is suggested that agent-based approaches have several advantages, such as the modelling of individual preferences of agents. Moreover, in the airline public transport area, Mao et al. [31] realize the need for short-term operational planning and scheduling methods in airport resource scheduling, because of unforeseen environmental influences. Their proposed agent-based system penalizes agents for decommitment and employs an auction system for task scheduling. As well as in [4] agents report costs, which could be seen as bidding, for taking over tasks and the agent with the lowest cost 'wins' the auction. In the MACS framework ants report the cost of advancing to the next bus stop and the ant with the lowest cost is chosen to advance. The cost function used to calculate the cost of taking on the next task is explained in section 3.3.

2.3 Disruption prevention

A more pro-active way of dealing with disruptions is to prevent them by making schedules more robust, and subsequently less sensitive to disruptions. Clausen et. al. [11] mention this disruption management through robustness specifically for airline scheduling, but these methods can of course be generalized to other forms of public transport, such as bus and train services. The main idea is to make the schedule more flexible to be able to absorb a disruption, or to add different types of recovery actions to facilitate an easy recovery. The first type is *absorption robustness* where the goal is to keep the plans feasible in case of smaller disruptions and avoid knock-on effects. The knock-on effect occurs when a driver or conductor of a delayed train needs to service a next train that will now also be delayed because it can only depart once the crew member of the delayed train has arrived. The second type, *recovery robustness*, aims at designing rotations and schedules in such a way that the plans fit well to the existing recovery strategies when a disruption occurs.

The disruption management process can be aided by combining the crew rescheduling task with small timetable adjustments. Delaying the departure of a train with a few minutes gives more flexibility to the rescheduling of crew duties. This is called *retiming* and it is clear that passengers prefer a train that is delayed a few minutes over a cancelled one [35].

Considering these approaches to prevent disruptions having an impact on the current timetable lies outside the scope of the current study. However, taking into account these pro-active measures can aid in creating more robust timetables that can withstand smaller disruptions and minimize knock-on effects when a timetable does become infeasible.

2.4 Global versus local search approaches

In local search approaches the direct vicinity of a crew member or vehicle is taken into account in creating a schedule. Moreover, in many cases a very short time frame into the future is considered in choosing a next task. This enhances the speeds with which solutions can be created, but has an adverse effect on the overall quality of the best solution obtained. A global search approach takes into account a very large part (or the entire) search space and tries to devise a schedule where the crew member's locations and work hours are taken into account when creating an overall schedule. This will ultimately increase the quality of solutions, but at the cost of increased computation time. To be able to get the best of both worlds; high quality solutions in a short time, good heuristics and a Multi-Agent System (MAS) approach will have to be considered. Multiple agents can communicate in order to create a schedule where different employees are not working against each other whereas heuristics can be put to work in searching for the next task in a relevant part of the search space.

The current system that NSR Logistics Innovation has developed [3] uses a MAS approach where the driver agents engage in a task exchanging process to improve the current schedule. Although their system outperforms a human dispatcher, it has no mechanism to test for an overall best solution. The MACS solution that we propose does make use of a global performance measure to check the quality of an entire schedule, while also ensuring that every ant (employee) makes choices about which task to perform next that will result in a local low cost for each individual ant.

2.5 Ant algorithms in public transport scheduling

Ant algorithms are optimization algorithms inspired by collective foraging behaviour of ants (i.e. finding the shortest path to a food source by depositing pheromone on that path) [17, 20, 15]. Foraging ants deposit pheromone on the paths that they travel. In Figure 2.1 this process is described step by step. First, in (a) ants arrive at a junction where they have an equal probability of choosing either the short path or the long path. Then in (b) and (c) ants start travelling over both paths whilst laying down pheromone in the process. As this continues the choice at the junction becomes more and more influenced by the amount of dropped pheromone. As can be seen in (d) the amount of pheromone on the short path has increased more than on the longer path. This happens because ants that choose the short path arrive at the next junction faster. Finally, the amount of pheromone on the short path is high enough to influence all ants to choose the short path.

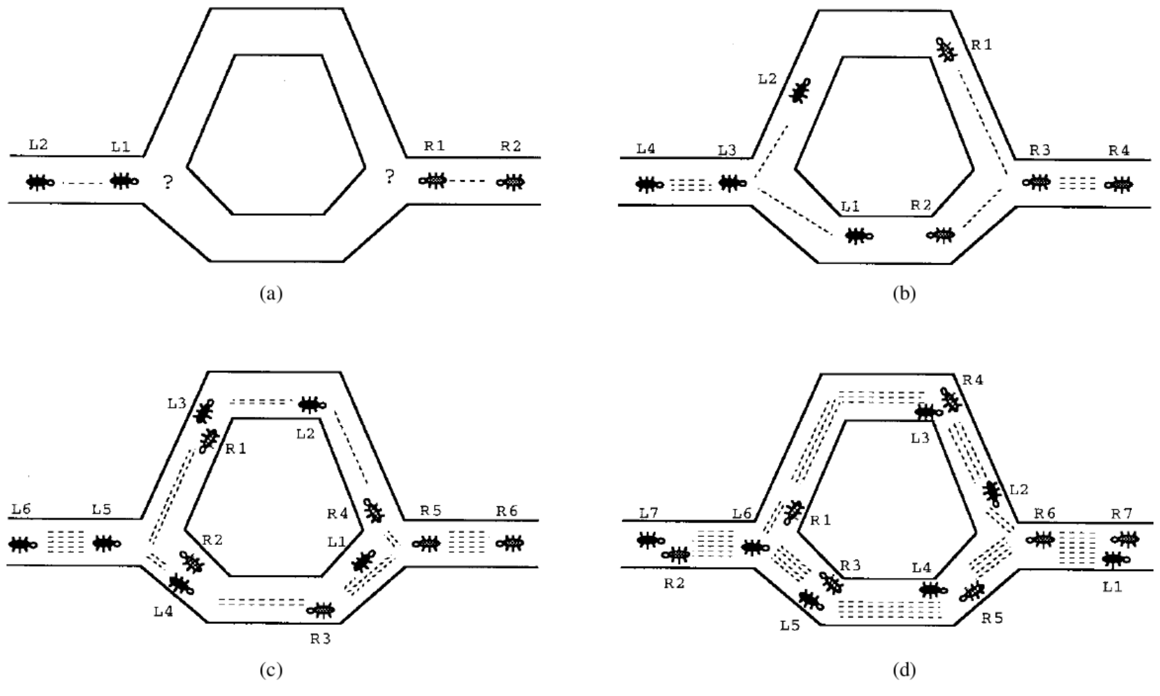

Figure 2.1: Pheromone deposits by foraging ants from Dorigo et al. [19].

The Ant Colony System (ACS) is an example of an Ant Colony Optimization (ACO) algorithm [16] and was developed to improve various optimization problems of ant algorithms consisting of individual artificial ants [19]. In ACS ants form a solution by individually building a complete tour (e.g. a solution to the Travelling Salesman Problem) preferring cities that are connected by short edges containing a high amount of pheromone. After each successive iteration the pheromone deposited on the best route found so far will be increased slightly and guide the ants in the next iteration towards a better route. Although most of the research on Ant Systems (AS) has been conducted on the Travelling Salesmen Problem (TSP) [20, 19, 21], where the shortest route between N cities that are visited exactly once needs to be found, there has also been research on scheduling in the public transport sector using AS.

Forsyth and Wren [25] use the Ant System (AS) to schedule bus drivers and mention that, using AS, optimality can only be guaranteed for very small problems. This suggests that AS alone is not sufficient to tackle scheduling problems for situations in the field without extensions, such as heuristics. Moreover, Bräysy and Gendreau [5] present a survey on research pertaining to the Vehicle Routing Problem with Time Windows (VRPTW), which can be described as the problem of designing least cost routes from one depot to a set of scatter points, that could be solved with an ACO inspired system. Montemanni et al. [33] propose to use the Ant Colony System (ACS) on a Dynamic Vehicle Routing Problem (DVRP), where a DVRP can be seen as a sequence of static Vehicle Routing Problems (VRP). Each static VRP is solved in a sequential fashion after which pheromone deposits are used to propagate information about the best route to the next VRP in the sequence.

Ant Colony Optimization (ACO) has not only been used for scheduling vehicles and crew on the ground but also in public air transport situations. Chih-Chung and Guang-Feng [10] use Ant Colony Optimization (ACO) based on the Ant Crew Scheduling Problem (ACSM) to solve an airline crew scheduling problem. They report that their model performs better on real world cases under the scheduling constraints and cost settings than previously developed Set Covering Problem based scheduling algorithms. Crawford et. al. [12] show a Hybrid Ant Algorithm on a Crew Pairing Problem in airline management operations. A crew pairing is a sequence of flight legs for an unspecified crew member starting and finishing in the same city. An ACO is used with Constraint Programming techniques like Forward Checking and shows promising results on a Crew Pairing Problem.

2.5.1 Multiple Ant Colony Systems (MACS)

The research described in the last part of the previous section gives rise to the use of Multiple Ant Colony Systems, which were first used by Gambardella et al. [29] on a Vehicle Routing Problem with Time Windows (VRPTW), in which different ACSs successively optimize multiple objective functions. Each ACS contains n ants that are homogeneous, while different ACSs can represent different types of ants. Gambardella et al. [29] use different types of ants, the first colony of ants minimizes the number of vehicles while the second colony minimizes the travel distances. Here the pheromone deposits are used to communicate the best routes between the colonies.

In a more recent study by Gajpal and Abad [28] a MACS was applied to a Vehicle Routing Problem with Backhaul (VRPB), where there are linehaul or delivery customers and backhaul or pick-up customers, which need packages delivered and picked up respectively. This is modelled on a graph on which vertex 0 represents the depot from where all vehicles start and has a demand of 0, whereas other vertices have different demands according to deliveries and pick-ups that have to be made. The difference in demand (packages for customers) and supply (capacity in vehicles) needs to be equalized to come to a feasible solution.

The MACS algorithm developed by de Jong and Wiering [13], where multiple Ant Colony Systems (ACS) cooperate to solve the Bus Allocation Problem (BAP), will be used as a starting point to solve the crew re-scheduling problem at the NS. A BAP arises when trying to construct m bus lines, each consisting of a sequence of bus stops. In a MACS each bus line is represented by an ACS, which means that the pheromone levels of each ACS are updated separately. Given a BAP consisting of n bus stops and m bus lines a MACS consisting of m ACSs will be initialized. Every ACS has an equal number of ants r, which results in r solutions (each consisting of a valid set of m bus lines) after each iteration. In Figure 2.2 the schematic representation of a MACS can be seen. Here all the nth ants of each colony together form one solution.

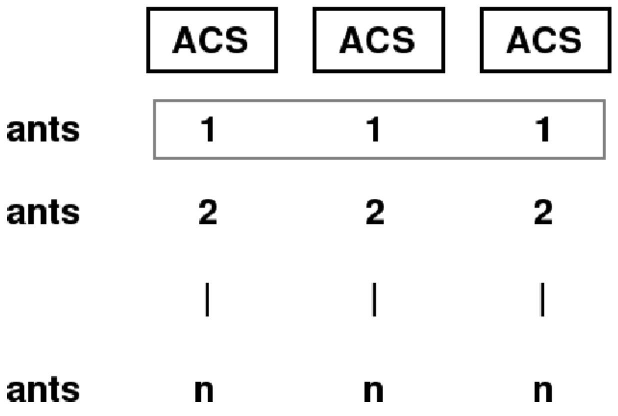

Figure 2.2: Multiple Ant Colonies working together.

The standard MACS approach will work on relatively small and simple problems with respect to constraints and search space size. The BAP problem, for example, consists of a few bus stations, a main bus station, a few bus lines, and the constraint that every bus station has to be visited exactly once and the main bus station has to be visited by every bus line. A proof-of-concept of our scheduling system using MACS on a small crew scheduling problem with a predefined train timetable showed that without additional search heuristics a solution of sufficient quality could not be found, because of the extra constraints in a crew scheduling problem. The more complex problem found at the crew scheduling stage of the NS will need an extended version of MACS to be able to generate solutions of a high enough quality within a reasonable time frame. The extended version of MACS is discussed in the next chapter.

Chapter 3

Methods

Currently the research group NSR Logistics Innovation is working on solutions to the scheduling and re-scheduling problems using Multi-Agent System (MAS) approaches (for example, [3]) where complex agents engage in task exchanges to increase the overall quality of the schedule. We, however, will be using an approach inspired by colonies of foraging ants which contain simple agents, and compare the solution to that of the MAS approach of the NS.

In order to get an initial feel for how well MACS would perform on a crew scheduling problem we built a proof of concept working with fictive data containing a few train stations, with a few lines, and a timetable on which crews have to be scheduled. Although the results of this proof of concept were promising, its solutions could be found easily by hand. This indicated that when adding extra constraints such as lunch breaks and increasing the number of possible tasks from 16 to more than 10000, MACS in its current form would not perform adequately. From the proof of concept we move on to Prototype Alpha which is discussed in chapter 4, that makes use of the NS Enschede dataset (section 3.1) supplied by NSR Logistics Innovation. The MACS algorithm and the additions made that led to MACS$^+$ are discussed below, as well as the various constraints (section 3.3.1) in the NS crew scheduling problem.

3.1 NS Enschede dataset

The initial proof of concept worked with a small dataset which contained four train stations. Between these stations 20 trains would depart at different times over a period of four hours. This dataset was sufficient for a first proof of concept to show if a MACS approach would be able to find a solution to a simple problem. To increase the complexity of the data a dataset based on a real life timetable was needed.

The research group at NSR Logistics Innovation provided us with a case study which contained such a dataset. Moreover, a solution for one base station (the Enschede train station) was also supplied to be able to test the quality of our solutions. The dataset is built up around tasks that have to be completed over the time of one workday. To be able to go from one task to another there are connections between tasks. Furthermore, there are also connections between base stations and tasks and vice versa. A base station is a train station from where an employee (conductor or train driver) will depart at the beginning of a shift. At the end of the workday this employee has to be back at that same base station.

A task has the following properties. Every task has a start point and end point at a train station. Furthermore, a task has a start and end time, from which the task's duration can be calculated. Lastly, a task has an id with which it can be retrieved easily.

A connection contains the task ids of the tasks between which the connection is placed, as well as the time needed to travel from task to task, which is denoted as the duration in minutes. An important aspect is the cost of travelling over the connection. This can for example be the cost of deadheading towards a next task or the waiting time before the next task begins. A connection also contains information about the possibility of taking a break, denoted with a time interval in which the break may be held. There is no possibility of a break if the interval is zero minutes. Apart from connections between tasks there are also connections between base stations and tasks and vice versa. These contain the same aspects as the other connection types. A base is located on a train station and has a separate id with which it is connected to task ids. This enables ants to choose connections leading towards their base station to end their shift properly.

To implement the supply and demand differences on the train stations tasks need to be grouped by start station. A station contains a list of the tasks that depart from that station. With this information one can calculate the demand for this station from now up until a certain point in the future. Furthermore, the amount of ants currently located at the station will provide the supply needed to service the tasks. If there is a higher demand than supply more ants will be attracted to that station to be able to handle all the tasks.

While extracting the tasks from the NS Enschede solution provided in the NS Enschede dataset it was found out that there were four duties being performed that were not contained within the tasks database. Without knowledge of the duration of these tasks and therefore the duration of entire solution an exact comparison can not be made between our results and the result provided by the NS. The four tasks are taken from the solution and replaced with idle time this enables us to give an indication of the quality of our solution compared to the solution provided in the NS Enschede dataset.

3.2 MACS$^+$

The basic MACS [13] is extended by adding a visibility heuristic to be able to enhance the quality of a solution while still maintaining a usable computation time. The computation time is less important in a scheduling setting where a schedule can be created days or even weeks in advance, whereas in a re-scheduling situation after a disruption very short computation times ($<$ 1 minute) are needed to be able to use the schedule immediately. The heuristic will work with supply and demand differences, as seen in [28], and aims to equalize the difference between these two values. Supply represents the number of crew members available to service a certain task (a train travelling from station A to B). Tasks represent the demand at one train station that needs to be satisfied. Equalling these two values will ensure that all tasks are executed and trains will run as scheduled.

This heuristic does not solve an important constraint stated in the crew scheduling problem;

the crew members have to be back at their departure station at the end of a shift. One can not implement a heuristic that will guide the ant (crew member) always to his/her departure station, because then the ant would never leave the departure station at the beginning of a shift. The probability that the ant will choose a task that will take it in the direction of the departure station should increase as the shift is nearing its end. This can be achieved by using a Gaussian probability distribution which represents a certain urge to go back that increases over time as the shift nears its end.

Each ant colony within the MACS will consist of r homogeneous ants. However, each colony can represent two different types of crew member, because there are both train drivers and conductors in the railway crew scheduling setting. Thus, colonies consist of either train driver ants or conductor ants. These different types of colonies working together were inspired by the use of different cooperating ant colonies [29].

The visibility heuristic will be used when an ant needs to choose the next task to fulfil. With the original MACS only a depth of one node is explored for a next step to take. Within the BAP problem this tactic is sufficient. However, in the NS crew scheduling problem there is the necessity to look at a greater depth in the not fully connected graph to ensure that the next node chosen leads to a path that has a low cost, even though the next node might have a higher cost than others, as the long term path needs to be considered. A low cost path is achieved by reducing the idle time of employees between tasks and the number of deadheading trips in a shift, as well as making sure that the employee is back at his base station at the end of his shift.

During the construction of a duty, ants will choose the next task with the lowest cost with respect to connection cost and the start time of the task compared to the current time. This decision-making process is influenced by the level of pheromone present on the connections; the higher the pheromone level the higher the likelihood of an ant choosing that connection towards the next task. Moreover, the aforementioned demand and supply differences also influence the decision making process. A higher demand at the train station where the next task will start, will give a greater likelihood of choosing the connection towards that task.

The duty cost, or rather the accumulated cost of executing tasks and travelling over connections between tasks, consists of several variables. To be able to compare our results to the results supplied by NSR Logistics Innovation we will use the same scoring mechanism for our solutions. A solution consists of duties, which are consecutive tasks completed during a work day. The cost of each duty consists of a base value to which the duration of the duty and the total cost of all connections is added, which results in a cost per duty. The total cost of a solution is calculated by summing over all duty costs and dividing by the number of duties. See the next section for the specifics of the MACS$^+$ model.

3.3 MACS$^+$ model

The MACS algorithm described by de Jong and Wiering [13] was developed to create bus lines between bus stops. In the NS crew scheduling problem the scheduling will be done in the opposite direction, since there are crews that need to be scheduled on an existing train timetable instead of creating the train timetable itself. Moreover, we have other constraints that will have to be satisfied, such as maximum working hours, break times, and returning to the start station (section 3.3.1).

The MACS$^+$ algorithm will use the NS Enschede dataset which contains n tasks (trains) on which m employees need to scheduled. Each employee is represented by an ACS and can perform multiple tasks. This translates to initializing a MACS with m ACSs, containing an equal number of ants r, thus giving us r solutions (each consisting of a valid set of m employee shifts) at the end of each iteration. The dataset also contains one solution for one base station (the Enschede train station) which contains 36 employees. Therefore, the number of employees will be set to $m = 36$, and as such the number of ACSs will also be restricted to 36.

The train timetable will be modelled as a connected graph $G = (V, E)$ where V is the set of vertices (each vertice represents a task) and E the set of edges, which represent the connections between the tasks. A very small example of the graph can be seen in figure 3.1. Here the yellow nodes (numbers 1 and 8) denote the same base station from where ants depart and arrive. Time progresses from left to right (i.e. node 1 is a task that starts earlier than node 4). It can be seen that taking a path containing nodes 1, 4, and 5 would probably result in more idle time as opposed to taking a path with nodes 1, 2, 3, 6, and 7 in the same time frame. Therefore, the latter path should give a lower cost, because more tasks were performed in the same time frame. An edge possesses information about its length and amount of pheromone of each separate colony. The length of an edge is affected by the time needed to reach the next task.

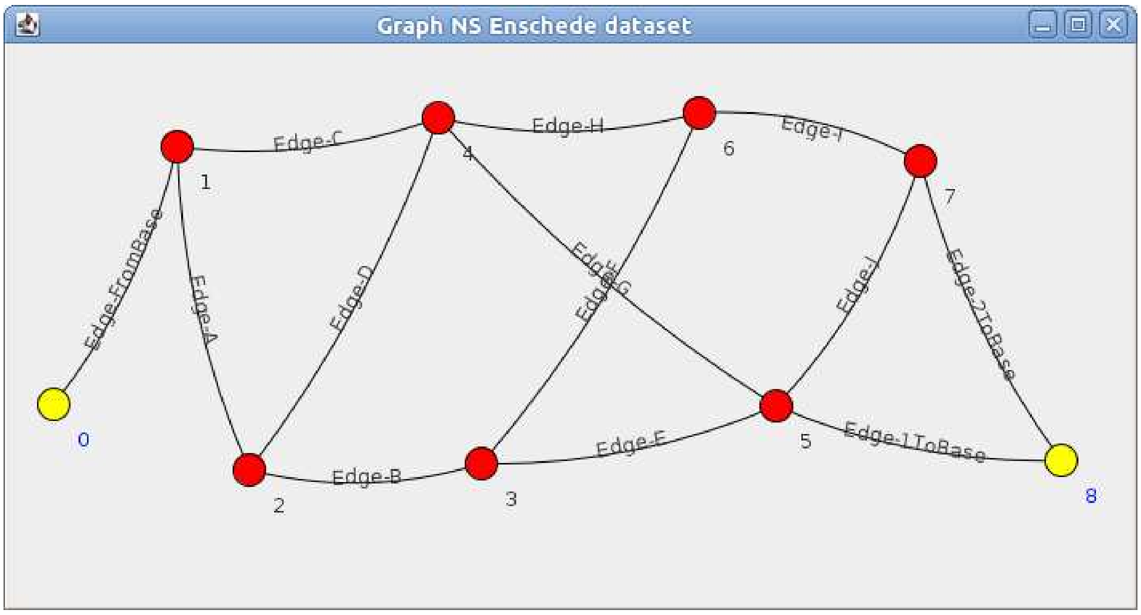

Figure 3.1: Example of the structure of the dataset

An ant will contain some information about the employee it represents. Every ant can represent either a driver or conductor. The id of the base station of the employee (ant), which is needed to be able to know where the employee will need to be at the end of a shift, is also contained within the ant. Apart from those values the moment at which a break is needed is also calculated for each ant. After a certain amount of hours a driver or conductor needs a break within a time interval, depending on the shift length and the moment in the day the shift is run. If this interval is reached the ant will be 'taken out' of the system at an appropriate station and be introduced again after its break is finished. Breaks can only be had at stations where there are canteen facilities. The remainder of this section explains the MACS model as described in [13] along with the changes made to use it with the NS Enschede dataset.

The visibility of an edge is denoted by η_{ij}^{l}, which represents the inverse of the cost of traversing that connection. Equation 3.1 below describes how η_{ij}^{l} is calculated and is adapted from [13].

$$\eta_{ij}^{l} = \frac{1}{b_{cost} + l_{cost}^{ij} + t^{ij} - d_{l}^{ij}} \tag{3.1}$$

b_{cost} denotes the base cost of taking a connection, which is set at 100. This is purely a precaution to avoid divisions by zero, because some connections have zero costs, especially connections from crew bases to a task and vice versa. To this base cost l_{cost}^{ij} is added, which denotes the extra costs that can result from taking a connection (for example; extra cost due to deadheading). Then t^{ij} is added, which represents the time difference between now and the start time of the task reached when taking this link. Finally, d_{l}^{ij} denotes the current demand for the next 30 minutes on the train station from where the next task starts, which is the number of tasks starting at that train station within 30 minutes minus the number of ants already present at the station.

Tasks that start beyond the search horizon of 60 minutes are ignored, as choosing a task too far into the future would result in too much idle time. Using a search horizon of less than 60 minutes was not considered as this would increase the possibility of ants not finding any tasks at all. Moreover, if the amount of tasks found is very low (< 5) the search space is not explored enough to consider possible better solutions. If there are no tasks within a horizon of 60 minutes this is extended to 120 minutes to search for more available connections. If there are still no possible connections to take the ant is assumed to have arrived in a sink node and as a result his shift will end after the current task is completed.

The amount of pheromone on an edge is denoted by τ_{ij}^{l}, which is the pheromone currently placed on the edge between task i and j for colony l. When an ant needs to choose a next task to execute it uses Equation 3.2 (or Equation 3.3 when $q > q_0$) to decide what the next task (N) will be. J_k denotes the number of tasks that are reachable within the search horizon and are still unvisited.

$$N = \begin{cases} \arg\min\limits_{h \in J_k} \{[\tau_{ij}^{h}] \cdot [\eta_{ij}^{h}]^{\beta}\} & \text{if } q \leq q_0 \text{ (exploitation)} \\ S & \text{otherwise (biased exploration)} \end{cases} \tag{3.2}$$

q is a random number ($0 \leq q \leq 1$) and q_0 a parameter ($0 \leq q_0 \leq 1$) that determines the importance of exploitation versus exploration. A high value of q_0 will result in more exploitation while a low value will increase the probability of biased exploration. When q is larger than q_0 the next train will be chosen randomly according to the probability distribution in Equation 3.3. Here the connections that have a lower cost will receive a higher probability of being chosen, hence the biased randomness instead of a purely random choice.

$$p_{ij}^{kl} = \begin{cases} \dfrac{[\tau_{ij}^{l}] \cdot [\eta_{ij}^{l}]^{\beta}}{\sum_{h \in J_k} [\tau_{ij}^{h}] \cdot [\eta_{ij}^{h}]^{\beta}} & \text{if } l \in J_k \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

To control the importance of the visibility β $(0 \leq \beta \leq 1)$ is used. When β is set to 0 the ants will not prefer short edges but solely rely on the amount of pheromone. The default value of β is 1 to denote equal importance to visibility and pheromone.

During the process of building a shift each ant reduces the pheromone level on the connections taken in order to increase the probability of a next ant choosing a different connection thereby making sure that there is more exploration in the region of the best solution found so far. Equation 3.4 below gives this local update for an edge (connection).

$$\tau_{ij}^{l} = (1-\rho) \cdot \tau_{ij}^{l} + \rho \cdot \tau_0 \tag{3.4}$$

The parameter ρ $(0 \leq \rho \leq 1)$ represents the evaporation rate of pheromone on a certain edge during a local update. τ_0 denotes the default level of pheromone given to each connection when initializing the MACS^{+} system. Before there has been a global update the local update will not influence the phermone level while the default pheromone value is equal to the current pheromone value.

At the end of each iteration a global update is performed which increases the pheromone on the edges belonging to the global best solution while pheromone on all other edges is decreased. The global update is performed using the equations shown below (Equation 3.5 and 3.6).

$$\tau_{ij}^{l} = (1-\alpha) \cdot \tau_{ij}^{l} + \alpha \cdot \Delta\tau_{ij}^{l} \tag{3.5}$$

$$\Delta\tau_{ij}^{l} = \begin{cases} (6500/L_{gb}) & \text{if edge } l \in S_{gb} \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

The parameter α $(0 \leq \alpha \leq 1)$ represents the learning rate. The global best solution is denoted by S_{gb} and the value of this solution is denoted by L_{gb} (Equation 3.8). The values of L_{gb} with the NS Enschede dataset lie in the neighourhood of 13000 in total costs for a solution. The value of 6500 was chosen to arrive at a $\Delta\tau_{ij}^{l}$ of around 0.5, which increases as the cost of solutions goes down. On the edges belonging to S_{gb} pheromone will be increased while the pheromone levels on all other edges will be decreased. Pseudocode for the MACS$^+$ model can be seen in listing 3.3.1.

Algorithm 3.3.1: MACS$^+$($parameters$)

```
for $iterations$
  for each $antNr \in nrOfAnts$
    while !DONE($ants$)
      for each $acsId \in nrOfAcss$
        $ant \leftarrow$ GETANT($antNr, acsId$)

        if $ant == done$
          then $continue$

        $linkMap \leftarrow$ GETLINKS($ant$)

        if NEEDSTOGOBACK($ant$)
          then GOBACK($ant$)

        $q \leftarrow$ RANDOM($0, 1$)

        if $q < qZero$
          then $link \leftarrow$ GETBESTLINK($linkMap$) (equation 3.2)
               $allLinks \leftarrow link$
          else $link \leftarrow$ GETBIASEDRANDOM($linkMap$) (equation 3.3)
               $allLinks \leftarrow link$

      if $allLinks == 0$
        then $continue$

      $ant \leftarrow$ GETBESTANT($allLinks$)
      $link \leftarrow$ GETBESTLINK($allLinks$)

      MOVEANTFORWARD($ant, link$)
      LOCALPHEROMONEUPDATE($link$) (equation 3.4)
      UPDATEDEMANDSUPPLY()

  $bestSolution \leftarrow$ CHECKSOLUTIONS($ACSs$)

  GLOBALPHEROMONEUPDATE($bestSolution$) (equation 3.5, 3.6)

  RESETNODES()
  RESETACSS()

SAVEBESTSOLUTION()
```

3.3.1 Constraints

The constraints that have to be taken into account when trying to solve the crew scheduling problem with the policy of the NS in mind are shown in Table 3.1. These constraints are enforced throughout the shift building process and at the end of each iteration when all generated solutions are checked. The ant that is able to choose a task with the lowest addition to his overall cost during the shift building process will be able to add that task to his shift. The solution with the lowest overall cost according to the constraints taken up in the cost function (section 3.3.4) is the best solution found so far.

Name	**Short description**	**Prototype Alpha**	**Prototype Beta**
Base station	Each employee will start and return to a base station.	x	x
Idle time	The amount of time during a shift when employee does nothing.	x	x
Lunch break	Lunch break at a certain moment.		x
Limited serviceable lines	Train drivers can only service certain lines.		
Layover time	It is assumed that layover time between tasks is included in the connection duration.	x	x
Maximum shift time	A shift can be 9 hours long at most.	x	x
Deviation from original schedule	Minimal deviation from the original schedule is preferred.		
Deadheading	Travelling on a train as a passenger to get from one station to the next		x

Table 3.1: Constraints for the crew scheduling problem, each of which is further explained in the remainder of this section.

3.3.1.1 Base station

Each employee will start his shift at a base station, which will not only serve as a start point but also as the end point of a shift. Arriving at the base station at the end of a shift has a high priority, as not succeeding in this will result in taxi costs or possibly even hotel costs, which have to be avoided.

3.3.1.2 Idle time

Idle time costs money while no work is being done. When employees switch between trains (tasks) it is almost inevitable that there will be some time between tasks that will be spent waiting. This idle time needs to be minimized by finding the right connections between tasks and the succession of tasks that minimize the time spent waiting. Idle time consists of every minute an employee is not actively working on a task or deadheading towards another task. As a break is an integral part of a shift it is not added to the idle time of an employee.

3.3.1.3 Lunch break

A lunch break is needed after a certain amount of working hours according to Dutch union laws. Veelenturf et al. [35] use the following rule to which our system will also adhere. A break of 30 minutes at a relief point (i.e. a station with a canteen) is needed if the replacement duty (shift) is longer than 5:30 hours. Moreover, the time before and after the break must be less than 5:30 hours. In addition to that we will use a maximum of 9 hours per shift, including a 30 minute break.

3.3.1.4 Limited serviceable lines

Train drivers are restricted to only service trains on lines that they are familiar with. This limits the flexibility of train drivers and constrains the scheduling problem further. However, conductors do not suffer from this limitation and can work on every train. Since the NS Enschede dataset does not contain data about the different available routes this constraint remains unused in both Prototype Alpha and Beta of MACS$^+$.

3.3.1.5 Layover time

When changing trains, or tasks in this case, there is time needed to get from train A to train B. However, for purposes of simplicity it is assumed that in the NS Enschede dataset the layover time between tasks is taken up in the duration of each connection between tasks. The number of minutes between tasks therefore has to be at least the duration of the connection, otherwise the next task will have begun before an employee has reached it.

3.3.1.6 Maximum shift time

Due to the capabilities of the employees and union laws the maximum shift time is restricted to 9 hours. Within these 9 hours there has to be a lunch break of at least 30 minutes at an appropriate time interval (see section 3.3.1.3 for more details).

3.3.1.7 Deviation from original schedule

In order to facilitate a smooth transition to the newly formed schedule from the original, the differences between the two should be minimized. This not only makes it easier to return to the original schedule after a disruption has been solved it also minimizes the strain on passengers travelling on a line that has been interrupted by a disruption. As the focus will only lie on scheduling, where there is no original schedule to deviate from, this constraint remains unused in both Prototype Alpha and Beta of MACS$^+$.

3.3.1.8 Deadheading

Deadheading can be used to increase the flexibility of employees that are either stranded on a train station where there are no more unserviced trains departing or on a train station where the waiting time for an unserviced train will incur high costs due to idle time. When deadheading, an employee can travel on a train as a passenger and resume his shift at the next station where there are unserviced trains. Although deadheading will increase the flexibility of an employee, the number of deadheading tasks have to be kept at a minimum because the idle time will increase with each deadheading task.

3.3.2 Return to base station

To be able to return to the base station an employee should be heading in the direction of that station at an appropriate point in time during his shift. If an employee returns too quickly his shift will be very short, leaving too many tasks unfinished. Whereas deciding to return near the end of a shift could result in the employee not being able to return to his base station within the remaining time of his shift, which increases the possibility that he will not make it home without extra costs such as a taxi ride or an overnight stay in a hotel.

Since there is no direction information in the NS Enschede dataset (see section 3.1) it was necessary to know which connection or which tasks would lead in the direction of the base station. Dijkstra's algorithm [14] was used to calculate the shortest path to the base station from every task in the dataset. The cost of a connection is based on the connection cost and the duration of the connection. Once these paths are calculated each node in the path stores the next node towards the base station. This eliminates the need to constantly calculate the shortest path toward a base from the current node.

A situation could occur where the next node in the shortest path is already occupied by another employee (i.e. the task is already performed by another ant). When this occurs the ant will choose the next best node with respect to connection cost and duration time to get to that node. From there the stored next node in the shortest path can be accessed again.

As mentioned above there is the need for an increasing urge over time to return to the base station. This urge is represented by a Gaussian distribution ($f_g()$ in Equation 3.7), adjusted for the shift length of the current ant, which will increase sharply as a shift draws to an end, and will give an employee a bigger urge to return to his starting point.

$$urge \quad = \quad l_{shift} + ((f_g() \cdot \sigma^2) \cdot \text{-1}) \tag{3.7}$$

$f_g()$ returns the next pseudo-random, Gaussian distributed value with mean 0.0 and standard deviation 1.0. l_{shift} denotes the length of a shift, for which the default is 540 minutes (9 hours). The variance is calculated as follows: $\sigma^2 = \frac{l_{shift}/2}{5}$. The urge is represented as a time in minutes which can be compared to the current time of an ants shift, if the urge is higher than the current time the ant will return, otherwise the ant will continue choosing tasks according to the MACS$^+$ algorithm.

The urge to get back is essentially the probability that an ant will choose a task that will lead back to his base station. However, it may occur that there is very little time left to return at the end of a shift. At that point a task leading back to the base station has to be chosen, regardless of the probability. In this case the ant will no longer participate in the task choosing process and just use the shortest path back to his base station in order to arrive there on time before his shift ends. When an ant is on his return journey he will perform the tasks that will lead him to his base station.

3.3.3 Rush hour

With public transport there are two distinct peaks in the number of passengers per hour at the beginning of a workday and at the end. During these peaks there are more trains in service and thus more tasks to be fulfilled. An even distribution of shift start times would create a sub-optimal schedule with not enough employees at peak times and too many at dull periods.

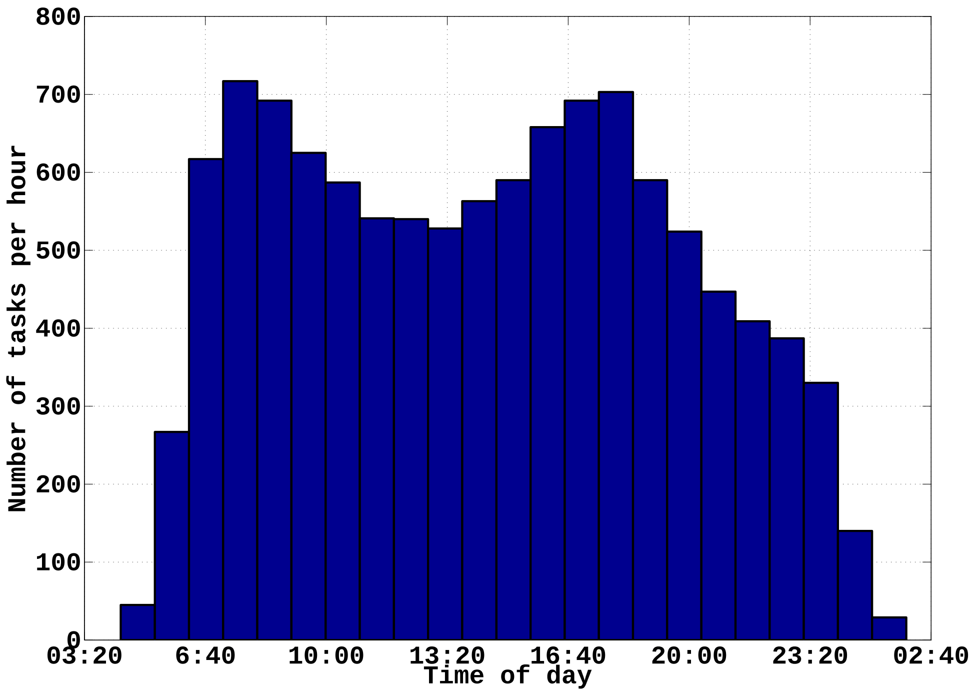

Figure 3.2: Departure times during one work day.

In Figure 3.2 it can be observed that in the NS Enschede dataset (see section 3.1 for details on the dataset) there is indeed an increase in the number of tasks per hour at the start and end points of a typical work day. We have therefore chosen to start four shifts per hour during rush hours and one shift per hour during the relatively dull periods. The first ant starts at 4:20 in the morning and from that moment on every 15 minutes a new ant is allowed to start until 7:30 when the morning rush hour is about to peak. The following ants start with an interval of 60 minutes until 15:00 when the afternoon rush hour begins until 18:00 in the evening when the rush hour peaks. The remaining ants, if any depending on the number of employees that will be used, will again start at 60 minute intervals. The ants starting their shift later in the day do not have the possibility to work 9 hours, because the last task starts at 01:59 in the morning.

3.3.4 Cost function

Where the original MACS algorithm applies the Average Travel Time (ATT) of people travelling from bus station a to b to find the best solution to the Bus Allocation Problem, we will use a cost function that calculates the cost of all duties in a solution. There are multiple constraints that affect the quality of the routes generated, some of which are mentioned above and are used to influence the process of creating a duty of tasks. The final solutions will be graded according to the following equation, where the duty of the j^{th} ant of every ACS is checked to obtain the value of one solution.

$$L_{gb}^{i} = \sum_{i} \frac{14400 + c_{ij} + (t_{ij} \cdot 60) + (10000 \cdot b_{ij})}{nrOfAnts} \tag{3.8}$$

The solution with the lowest value for this equation will be the best crew scheduling solution found so far. The total time in minutes of a duty is denoted by t_{ij} which is multiplied by 60, in accordance with the cost function used in the solution provided together with the NS Enschede dataset. c_{ij} denotes the total cost of the connections chosen. These two values are added to the initial duty cost of 14400. Finally, the number of ants that are not back at their base station (denoted by b_{ij}) are multiplied by 10000 which serves as a high penalty for not being back.

3.4 Development environment

To enable fast development we chose to write our software using Java 1.6 and Eclipse 3.5.2 [1]. This enabled us to write code quickly and produce prototypes fast. Moreover, because of the ease of use with respect to the graphical environment we were able to visualize the dataset of our proof-of-concept in a graph. This gave us insights that would not have been seen with a text-based output.

Migrating the source code to C++ could result in a decrease of computation time, but in light of the limited time for this project we did not work in that direction.

The test system used to run all experiments was an Intel Core 2 Duo E8400 running at 3 GHz combined with 4 GB of system memory. Although the test system contains a dual core processor the algorithm is single threaded and will therefore only use one core.

Chapter 4

Prototype Alpha

4.1 Introduction

In this chapter Prototype Alpha is presented. This prototype contains a subset of the constraints mentioned in section 3.3.1 to be able to run early experiments with the MACS$^+$ model. Prototype Alpha was developed to be used on the NS Enschede dataset described in section 3.1.

4.2 Implementation

Insights obtained from the research group at NSR Logistics Innovation led us towards first solving the problem of creating a crew schedule based on a rolling stock time table for one workday, as it became apparent that the re-scheduling problem would be significantly more difficult compared to normal scheduling for an entire day. The NS Enschede dataset contains a solution for one base station which forms a schedule for one day with 36 employees. The aim of this prototype is to generate a schedule for this base station and approach the quality of the given solution with respect to minimizing costs and computation time while maximizing the number of tasks performed and the number of ants back at their base station. However, the minimization of computation time is of lower priority than the other goals.

Name	**Short description**
Base station	Each employee will start and return to a base station.
Idle time	The amount of time during a shift when employee does nothing.
Layover time	It is assumed that layover time between tasks is included in the connection duration.
Maximum shift time	A shift can be 9 hours long at most.

Table 4.1: Constraints in the NS Enschede Case used in Prototype Alpha. See section 3.3.1 for a detailed explanation of each constraint.

In an effort to keep the scheduling problem manageable, Prototype Alpha will use a subset of the constraints mentioned in section 3.3.1. Table 4.1 displays the constraints taken into account in this prototype. It can be seen that the lunch break constraint, among others, has been omitted here for simplicity. However, every ant needs to be back at his base station, the idle time needs to be minimized as well as the layover time, and there is a maximum shift time of 9 hours.

4.3 Experiments

The experiments conducted with Prototype Alpha made use of the NS Enschede database supplied by NSR Logistics Innovation and were set up as follows. Three sets of experiments were executed, each of which concentrated on one particular parameter to be able to ascertain the influence of that parameter. Within each set there were three different parameter values that were tested. For each value of a parameter 10 runs were performed from which an average was taken (standard deviation and best values are shown in Tables 4.6, 4.7, and 4.8) which are shown in section 4.4. The default values for every parameter can be seen in Table 4.2 below. These values were found experimentally and result in a reasonable solution. The number of employees was kept at 36 in accordance with the solution of the NS Enschede database. The ant start times were according to the distribution described in section 3.3.3.

Name	Description	Value
iterations	The number of iterations the algorithm will run.	100
nrOfAcss	The number of Ant Colony Systems (ACS) that will be initialized. This is equal to the number of ants per ACS.	36
τ_0	The default value of pheromone which is placed on all links.	0.05
ρ	Denotes the evaporation rate of pheromone.	0.005
α	The learning rate influences the increase of pheromone on links belonging to the global best solution.	0.0025
q_0	Denotes the amount of exploration against exploitation. A higher value means more exploitation. A value of 0 indicates biased randomness (see Equation 3.2 and 3.3).	0.75

Table 4.2: Configuration parameters and their default values.

In the first set of runs the effect of adjusting the amount of exploration against exploitation was tested. Results from the experiments conducted by De Jong and Wiering [13] on all BAP problems with different numbers of busses show that a q_0 of 0.95 gave the best results. A low q_0 results in higher exploration, but without high exploitation the convergence to an optimum is far slower. q_0 was set to the following three different values as seen in Table 4.3

Parameter	Values		
q_0	0.95	0.75	0.25

Table 4.3: Different amounts of exploitation against exploration.

Pure biased exploration, with $q_0 = 0$, will result in biased randomness, whereas a value of 1 would mean pure exploitation without a search for different paths. Both pure biased exploration and pure exploitation are not suitable for finding an optimum in reasonable time. All other parameters were kept at their respective default values as shown in Table 4.2. The results of this set of experiments can be found in section 4.4.

The second set of runs were aimed at finding the influence of changing the learning rate α, which could of course greatly influence the outcome of the experiments. The values for this parameter can be seen in Table 4.4. The learning rate is crucial in making sure that the algorithm converges to an optimal solution. Setting this parameter too low will result in never finding the optimal solution, whereas setting it too high will result in fast convergence to a likely sub-optimal solution.

Parameter	Values		
α	0.05	0.0005	0.00005

Table 4.4: Different learning rates.

Finally, performance for different default pheromone levels (τ_0) was tested. As with the previous runs the other parameters remained at their respective default values as stated in Table 4.2. The values can be seen in Table 4.5. Here a balance has to be found between the amount of pheromone on each connection and the cost of taking that connection. If the pheromone level is too high the influence will overpower the reliance of the algorithm on connection costs and reach a local optimum too soon.

Parameter	Values		
τ_0	0.5	0.005	0.0005

Table 4.5: Different default pheromone levels.

4.4 Results

The following figures show the results from the experiments conducted with Prototype Alpha. In each image there are three graphs visible. The top graph displays the average cost of a shift in one solution. The middle graph shows the total number of tasks completed by all ants in one solution. Finally, the bottom graph shows the number of ants that were back at their base station at the end of their shift. For each value of the tested parameters there were 10 runs, the results of which were averaged to come to the graphs presented below.

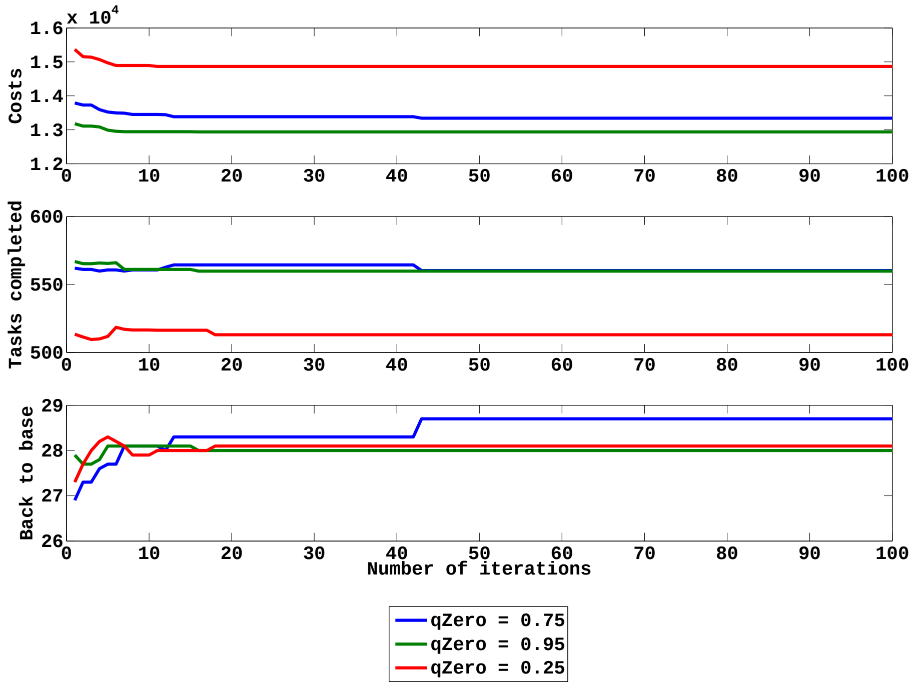

Figure 4.1: Experimental results with different values for q_0.

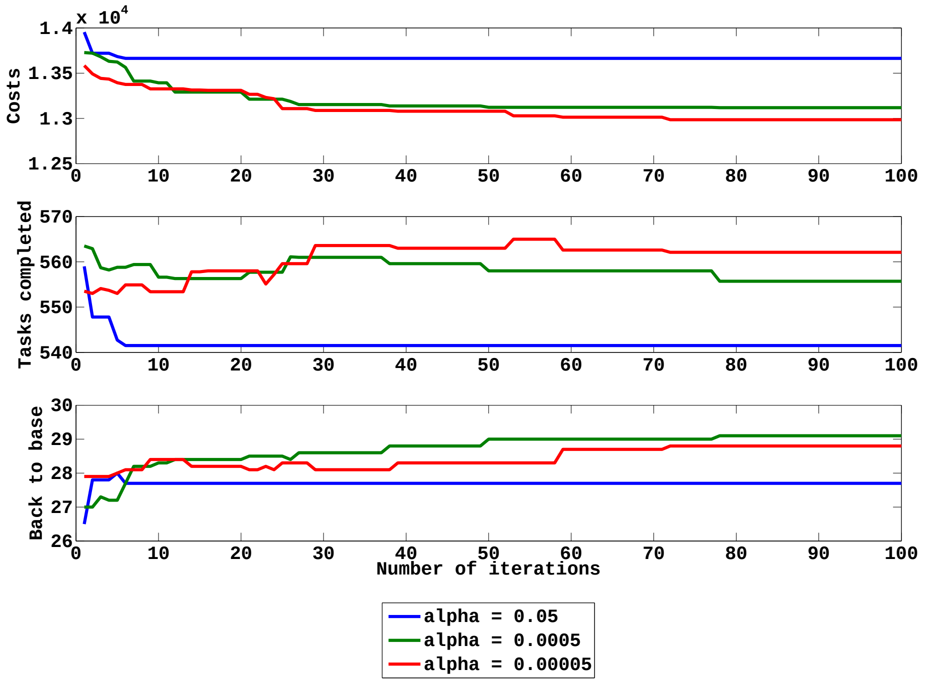

Figure 4.2: Experimental results with different values for α.

Figure 4.1 displays the results obtained when varying q_0, which controls the importance of exploitation versus exploration. It can be seen that a high amount of exploration ($q_0 = 0.25$) leads to a higher average cost when compared to more exploitation. Thus, a less optimal solution will be found when relying on high exploration.

Figure 4.2 shows the results obtained when varying the learning rate α between 0.05, 0.0005, and 0.00005. A high learning rate results in a fast convergence to a solution, which is likely to be a local optimum. A slower learning rate allows for more exploration before convergence to one solution, which leads to better solutions with respect to average costs, total number of tasks executed, and total number of ants back at their base station.

In Figure 4.3 (below) the results of differing the default pheromone level (τ_0) can be seen. The levels were set to 0.5, 0.005, and 0.0005 and again there were 10 runs for each configuration. When setting the initial pheromone level very low the total number of tasks performed drops while the number of ants that return to their base station stays relatively high.

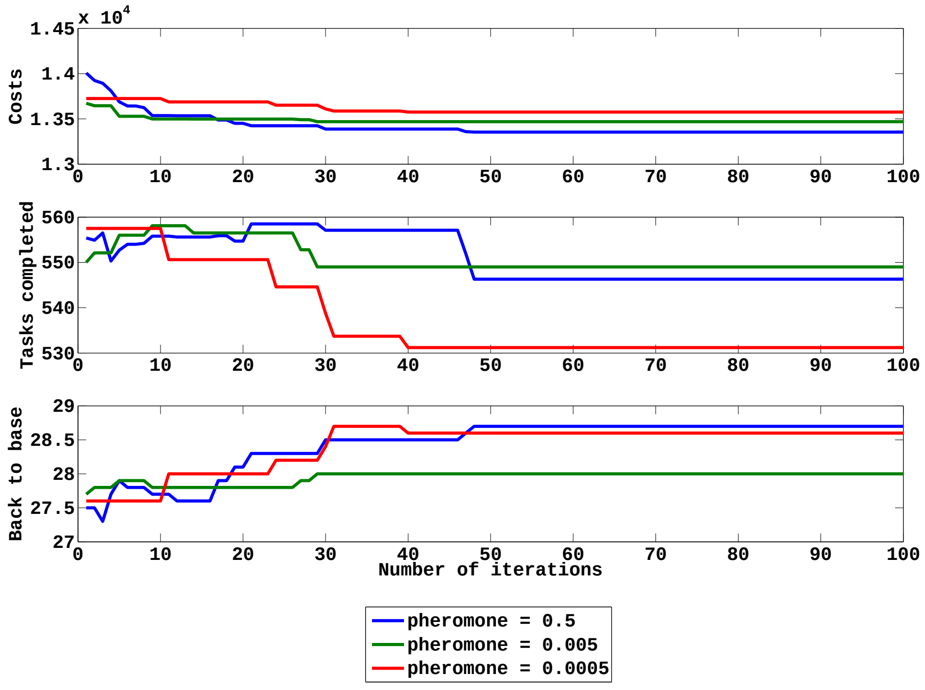

Figure 4.3: Experimental results with different values for τ_0.

The next three tables show the results again with the addition of the standard deviation between the individual runs and the best solution found for each parameter value. The best solutions found for each parameter are highlighted in bold and were selected on the basis of cost and the number of tasks executed.

			Costs	Tasks	Back
q_0	0.75	best	12800	583	31
		avg	13341	560.3000	28.7000
		std	304.3225	15.4564	1.1595
	0.95	best	**12063**	**583**	**32**
		avg	12938	559.8000	28.0000
		std	360.7676	15.0022	1.7638
	0.25	best	14409	549	30
		avg	14864	513.0000	28.1000
		std	230.1674	16.9771	0.9944

Table 4.6: q_0 values 0.75, 0.95, 0.25

			Costs	Tasks	Back
τ_0	0.5	best	**12823**	**582**	**30**
		avg	13354	546.3000	28.7000
		std	261.3819	18.8741	1.0593
	0.005	best	12857	580	30
		avg	13470	549.0000	28.0000
		std	481.4001	18.3787	1.1547
	0.0005	best	13118	576	31
		avg	13576	531.2000	28.6000
		std	278.5007	29.3969	1.5055

Table 4.7: Default pheromone (τ_0) values 0.5, 0.005, 0.0005

			Costs	Tasks	Back
α	0.05	best	13068	581	31
		avg	13663	541.5000	27.7000
		std	367.8583	19.8564	1.7670
	0.0005	best	**12654**	**583**	**30**
		avg	13119	555.7000	29.1000
		std	308.9057	14.4457	0.8756
	0.00005	best	12523	582	30
		avg	12986	562.1000	28.8000
		std	261.6996	15.1983	1.5492

Table 4.8: α values 0.05, 0.0005, 0.00005

4.5 Discussion

The performance differences for the different q_0 values were as expected. A very high amount of exploration ($q_0 = 0.25$) means that good solutions that are found are not exploited enough which leads to suboptimal solutions after 100 iterations. A value of $q_0 = 0.75$ returns the best result, when not only taking into account the average duty cost, but also the number of ants that have returned to the base station. Furthermore, a high learning rate ($\alpha = 0.05$) results in fast convergence to a solution that is a local optimum. It can be seen in Figure 4.2 that a learning rate lower that 0.0005 shows the best results after 100 iterations. The results for the different pheromone levels show that a higher default pheromone level gives a better result, indicating that the algorithm relies on pheromone to choose the best connections.

Comparing the results of experiments with different parameter values shows how these parameters influence the overall solution quality. However, there is no benchmark to test the overall performance of the system with only changing parameters. To be able to test the quality of our system we compared our results with a solution for one base station (the Enschede train station) provided by NSR Logistics Innovation. The number of tasks executed per solution, which lies around the 560 tasks per solution mark, comes in the neighbourhood of the 584 tasks executed in the NS Enschede dataset solution. However, when comparing the number of employees that have returned to their base station at the end of a shift there is a performance gap. In the NS Enschede dataset solution every duty ends at the base station, where our system, on average, does not come above 30 out of 36 employees back at base. This will result in high taxi costs, hotel costs, or train travel costs, which were not taken into account in calculating the total cost of the solution in Prototype Alpha.

Apart from the low number of ants that return to their base station when compared to the Enschede solution it can also be seen that the number of tasks does not increase over time. We would expect to see an increase in the number of tasks executed as time progresses. Moreover, in the experiments with varying default pheromone levels the number of tasks actually decreases instead of increases. The reason for this behaviour lies in the cost function (see Equation 3.8). In the cost function the total time of a duty is added to the total cost of a solution. Solutions with shorter duties have a lower cost according to the current cost function, although creating shorter duties is not our goal. The connection times between the tasks in a duty should be minimized, not the number of tasks. In the next chapter the changes made to the cost function to increase the number of tasks executed are discussed.

The low number of ants back at their base station and the number of tasks not increasing over time shows that changes have to be made to the MACS$^+$ model to increase the quality of generated solutions. Moreover, there were no experiments conducted to test the influence of the evaporation of pheromone (ρ) which will be taken up in the experiments conducted with Prototype Beta, which is discussed in the next chapter.

Chapter 5

Prototype Beta

5.1 Introduction

The experiments conducted with the Prototype Alpha version of MACS^{+} (see chapter 4) showed us that, although the results were promising, there was still room for improvement. Especially the constraint of returning to a base station at the end of a shift was not always met, which invalidated a solution altogether. Therefore not only the mechanism to return to the base station was changed, but deadheading was also introduced to make sure that ants would have more flexibility in returning to their base.

It was also found that due to the use of local search when choosing the next task the global consequences of executing a task were not taken into consideration. Ants did search n nodes deep to make a decision based on a future path. However, this was done for each ant individually, without considering what would happen if two or more ants would arrive at the same task. This limited the quality of the local search. Therefore, a look ahead function was created to be able to make a more weighted decision about which task to choose next, also considering which tasks the other ants would perform. In this function every ant will create a path of five tasks into the future using the MACS^{+} algorithm which is then scored and pheromone is laid down on the best path. Repeating this process n times will create a pheromone landscape that is then used to better guide the ant towards an overall optimal path instead of just a local optimum.

5.2 Methods and implementation

In this section we will discuss the various changes made to Prototype Alpha and, more importantly, why these changes were made.

5.2.1 Cost function

Initially the cost function was largely implemented according to the specifications used in the NS Enschede dataset (see section 3.1). This lead to the following equation.

$$L_{gb}^{i} = \sum_{i} \frac{14400 + c_{ij} + (t_{ij} \cdot 60) + (10000 \cdot b_{ij})}{nrOfAnts} \tag{5.1}$$

From the result of the experiments conducted with Prototype Alpha it became clear that the use of the total time of a shift $(t_{ij} \cdot 60)$ had a negative effect on the overall solution. The generated shifts tended to become shorter as the iterations progressed because this helped to reduce the value of the cost function. A shorter shift with the same amount of tasks is of course an optimization, but the reduced shift time was achieved through reducing of the number of tasks per shift. To make sure that the amount of tasks in a shift remained a priority, the following change was made to the cost function.

$$L_{gb}^{i} = \sum_{i} \frac{14400 + c_{ij} - (n_{ij} \cdot 100) + (b_{ij} \cdot 10000)}{nrOfAnts} \tag{5.2}$$

In equation 5.2 we can see that the total time of a shift $(t_{ij} \cdot 60)$ has been replaced by the number of executed tasks $(n_{ij} \cdot 100)$. n_{ij} represents the number of tasks executed from which the number of deadheading tasks are subtracted. This means that an increase in the number of tasks performed will decrease the cost of a solution. Deadheading tasks essentially become idle time and are discussed in the next section.

5.2.2 Deadheading

As explained in section 3.3.1.8, deadheading according to the NS is travelling in a train as a passenger instead of working on that train as either a train driver or conductor. Of course this is not something that should occur frequently as it results in more time spent not performing tasks during a shift. However, quite frequently employees could find themselves in a sink node, because there are no unserviced tasks departing from that station. To be able to depart from such a station the ant is allowed to travel on a train as a passenger to a next train station in order to continue working from there. By allowing deadheading the overall number of tasks executed and the probability of ants being able to finish their shift at a base station will increase, because they will not find themselves in a sink node.

5.2.3 Look ahead function

Within Prototype Alpha the choice of which task to execute next was always limited to a local search into the directly adjacent tasks. Although this search was guided by pheromone levels that changed over time due to a global pheromone update, the choice involved local information. As a result the effects of accepting a particular task later on in the shift were not fully taken into account. Ants did search n nodes deep to consider future nodes. However, this did not take into consideration the possibility of other ants coming across the same tasks. To increase the consideration for a global optimum a look ahead function was created which is based on a Monte-Carlo Tree Search (MCTS) algorithm developed by Chaslot et al. [9].

Look ahead functions have already been successfully applied for Ant Colony Optimization by Gagne et. al. [27] and Michel and Middendorf [32].

MCTS is a best-first search algorithm which is based on randomized exploration of the search space and is tested on the game of Go [9]. Since there is no knowledge about good moves in the beginning, search is performed randomly. As the game progresses more knowledge about good moves becomes available as a result of exploring the search space and the algorithm becomes more accurate in predicting promising moves. In the MCTS algorithm nodes represent different states of the game and the algorithm consists of four steps. First the tree is traversed until a leaf node is reached, then one or more nodes are expanded. From there a simulated game is played and the result is back propagated towards the root node. This process is repeated as long as there is time left. Every time the result is back propagated nodes are strengthened with information about the outcome of a game played through this node. This eventually leads to choosing nodes that lead to playing an optimal game.

In our look ahead function we will use the playing of a simulated game by creating a partial solution for which a global pheromone update is executed. This process is repeated *m* times to create a sufficiently large difference in pheromone between connections to influence the choice for a next task on a more global level. In creating the partial solution the MACS$^+$ algorithm is used to create a separate field of look ahead pheromone levels. When the algorithm starts the look ahead function is called and is initialized with the normal pheromone. Now the look ahead is executed to change the look ahead pheromone levels. After the look ahead function ants will actually choose tasks to execute. This choice is influenced by the look ahead pheromone. This process is repeated until all ants have created shifts from which the best solution is chosen and the normal pheromone update is performed. Next this normal pheromone is again used to initialize the look ahead pheromone in the next iteration of the look ahead function. Pseudocode of the look ahead function can be seen in listing 5.2.1.

Algorithm 5.2.1: LOOKAHEAD($parameters$)

```
for $lookAheadIterations$
  for each $antNr \in nrOfAnts$
    comment: While there are ants that have not done $m$ tasks.
    while $!done$
      for each $acsId \in nrOfAcss$
        $ant \leftarrow$ GETANT($antNr, acsId$)

        if $ant == doneLookAhead$
          then $continue$

        $linkMap \leftarrow$ GETLINKS($ant$)

        if NEEDSTOGOBACK($ant$)
          then GOBACK($ant$)

        $q \leftarrow$ RANDOM($0, 1$)

        if $q < qZero$
          then $link \leftarrow$ GETBESTLINK($linkMap$) (equation 3.2)
               $allLinks \leftarrow link$
          else $link \leftarrow$ GETBIASEDRANDOM($linkMap$) (equation 3.3)
               $allLinks \leftarrow link$

      if $allLinks == 0$
        then $continue$

      $ant \leftarrow$ GETBESTANT($allLinks$)
      $link \leftarrow$ GETBESTLINK($allLinks$)

      MOVEANTFORWARD($ant, link$)
      LOCALLOOKAHEADPHEROMONEUPDATE($link$) (equation 3.4)
      UPDATEDEMANDSUPPLY()

  $bestPartialSolution \leftarrow$ CHECKPARTIALSOLUTIONS($ACSs$)

  GLOBALLOOKAHEADPHEROMONEUPDATE($bestPartialSolution$) (equation 3.5, 3.6)

  RESETNODES()
  RESETACSS()
```

5.2.4 Lunch break

As mentioned in section 3.3.1.3 the lunch break is a vital part of a shift. However, it was omitted in Prototype Alpha to keep the scheduling problem manageable. The possibility of taking a break is included in the NS Enschede dataset (section 3.1), where connections can have a breakStartTime and a breakEndTime. If these values are both zero there is no possibility of taking a break. Moreover, if there is the possibility of taking a break, there is always time for a break of at least 30 minutes.

The following rule from Veelenturf et al. [35] will be used in MACS$^+$ as discussed in section 3.3.1.3. A break of 30 minutes at a relief point (i.e. a station with a canteen) is needed if the duty (shift) is longer than 5:30 hours. Moreover, the time before and after the break must be less than 5:30 hours. In addition to that we will use a maximum of 9 hours per shift, including a 30 minute break. This means that within a 9 hour shift a break should start between 3 hours and 5:30 hours.

As soon as an ant arrives in that time interval only connections that contain a valid break will be chosen to ensure that the ant will be able to take a break at the appropriate time. There is no check done if the task lasts more than the minimum shift length of 5:30 hours, while currently every ant will try to work a shift of 9 hours including a break.

5.2.5 Back to base station

A completed shift of an employee is only seen as valid when, at the end of his shift, he is back at the base station from where he started his first task of the day. To ensure that ants would return to their base station we implemented an urge to go back (see section 3.3.2) that would increase as the end of a shift would come nearer. As can be seen in Equation 3.7, a Gaussian distribution was used to achieve an increase in urge as the end of a shift comes closer.

From the results of the experiments with Prototype Alpha it became apparent that this urge was ineffective in making sure that every ant would return to his base station. This is largely due to the fact that not only the time an ant has left in his shift is important, but more so the location of the ant in the dataset. If the ant has chosen a path that leads away from a base station a return path towards the base has to be taken far earlier then when an ant has chosen tasks that are relatively close to his base station. To solve this problem an extra value was added to Dijkstra's shortest path [14] described in section 3.3.2. In addition to the next node in the shortest path towards a base station a node now also contains the time needed to travel to a base through the current node. This cutoff point combined with the next node is now sufficient to make sure that ants return in time before their shift ends. Before every task is chosen a check is performed as to how much time an ant has left in his shift against the time needed to get back to his base from the current position. If there is not sufficient time to return after the next task is chosen the ant will instead opt for the next task that is contained in Dijkstra's shortest path.

This solution has made the urge to return obsolete and it has therefore been removed. Moreover, in combination with deadheading the number of ants returning to their base station is always 100%.

5.2.6 Shift start time

As described in section 3.3.3 there are two distinct peaks in the number of tasks that need to be performed per hour. These two peaks are the two rush hour periods on an average work day. To prepare for this increase in demand the start times of ants were such that during the rush hour periods four ants per hour would start a shift and one ant per hour would start a shift in the dull periods. We found experimentally that changing the time of the day where four ants per hour start a shift had a great influence on the overall schedule that was created. This lead us to extract the employee start times from the NS Enschede dataset solution and apply those to our system. This improved the overall schedule, which indicates that varying the start times can be beneficial in trying to find better solutions. The current research will not focus on finding optimal distribution of employee start times and keep making use of the NS Enschede datase shift start distribution. However, in future work the distribution of start times should be taken into account.

5.3 Experiments

As with Prototype Alpha described in chapter 4, the dataset used in these experiments was again the NS Enschede dataset. The first set of experiments were conducted without the lunch break constraint to be able to compare the results to the experiments with Prototype Alpha. The default values for this first set can be seen in Table 5.1. The number of iterations for each run has been reduced to 50 because with the introduction of the lookahead function the computation time increased significantly. Moreover, the number of runs for each configuration has been reduced to 5. Whereas the runs of 100 iterations with Prototype Alpha take around 30 minutes, depending on the configuration of the parameter values, the runs of 50 iterations with Prototype Beta will take around 4 hours to complete.

Name	**Value**	**Symbol**
Iterations	50	
Number of ACS's	36	
Default pheromone	0.005	τ_0
Evaporation rate	0.05	ρ
Pheromone deposit	0.025	α
Exploitation/Exploration	0.95	q_0

Table 5.1: Default values.

5.3.1 Experiments without a lunch break

The different configurations can be seen in Table 5.2. The difference between high exploitation and high exploration will be checked as well as the influence of high deposits of pheromone which essentially translates to a higher learning rate. Moreover, the influence of a faster evaporation rate will also be included in the experiments. This could improve exploration of the search space because older paths loose their pheromone faster if no ants travel over them.

The combination of high pheromone deposits and a high evaporation rate will increase the difference between well travelled connections and less travelled connections. Furthermore, the influence of a higher default pheromone level was also tested to find if the increased contribution of pheromone to connection costs could lead to better solutions.

Configuration	q_0	α	ρ	τ_0
Exploitation	**0.95**	0.025	0.05	0.005
Exploration	**0.50**	0.025	0.05	0.005
High learning rate	0.95	**0.25**	0.05	0.005
High evaporation rate	0.95	0.025	**0.5**	0.005
High learning rate & evaporation rate	0.95	**0.25**	**0.5**	0.005
High pheromone	0.95	0.25	0.5	**0.5**
High pheromone with exploration	**0.75**	0.25	0.5	**0.5**

Table 5.2: Different parameter configurations. (without a lunch break)

5.3.2 Experiments with a lunch break

In these experiments the lunch break constraint was added. Moreover, the default pheromone level (τ_0) was raised to 0.5 to have a greater influence on the cost of choosing a particular connection. Here we kept the evaporation rate equal over all configurations while changing the learning rate and the exploitation/exploration differences.

Configuration	q_0	α	ρ	τ_0
1	**0.95**	**0.25**	0.05	0.5
2	**0.75**	**0.25**	0.05	0.5
3	**0.95**	**0.4**	0.05	0.5
4	**0.75**	**0.4**	0.05	0.5
5	**0.75**	**0.6**	0.05	0.5

Table 5.3: Different parameter configurations. (with a lunch break)

5.4 Results

5.4.1 Results without a lunch break

Here the results for the experiments with the final version without the lunch break are presented. The results for different parameter values are shown in one graph (Figure 5.1) while the best result of MACS$^+$ is shown in combination with a Greedy and a Random algorithm in Figure 5.2. Since in all solutions every ant is back at his base station at the end of a shift the results for the number of ants that have returned will be omitted here. As with the experiments performed with Prototype Alpha the standard deviation and the best solution for every configuration are shown in a table (Table 5.4).

In Figure 5.1 it can be seen that a high amount of exploration ($q_0 = 0.5$) has a negative influence on the quality of the final solution. Using a higher learning rate ($\alpha = 0.25$) or a higher evaporation rate ($\rho = 0.5$) will increase the solution quality, while both a higher learning rate and a higher evaporation rate will give even better results. A higher default pheromone level ($\tau = 0.5$) will lower the cost of a solution even further and give an increase in the number of tasks completed.

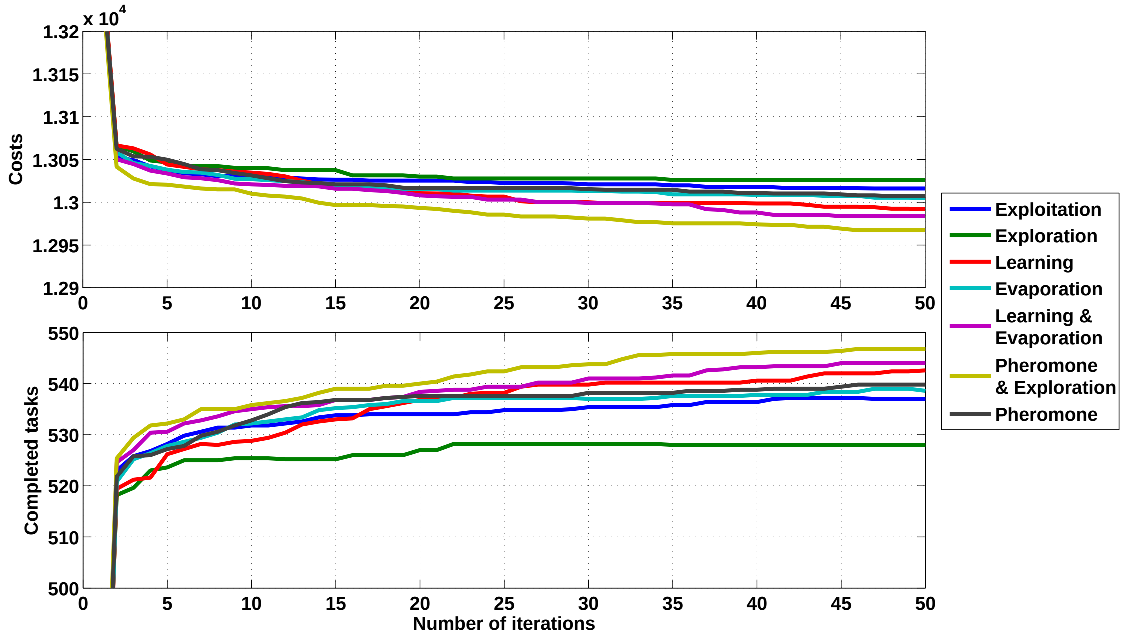

Figure 5.1: Experimental results for different configurations.

In Table 5.4 it can be seen that, although the average solution quality increases the standard deviation goes up. With a higher standard deviation the probability that future solutions will lie further away from the current average will increase. Solutions found could be far worse than the solution found for a parameter configuration that performs less but has a far lower standard deviation.

		Costs	Tasks	Back
Exploitation	best	13010	543	36
	avg	13016	537	36
	std	**6.7757**	4.4721	0
Exploration	best	12993	532	36
	avg	13026	528	36
	std	24.8917	**2.5495**	0
High learning rate	best	12960	547	36
	avg	12992	542.6	36
	std	19.7007	3.5777	0
High evaporation rate	best	12990	544	36
	avg	13005	538.6	36
	std	15.7625	5.8138	0
High learning rate & evaporation rate	best	12957	550	36
	avg	12984	544	36
	std	15.5407	3.6742	0
High pheromone	best	12968	550	36
	avg	13007	539.8	36
	std	25.4588	6.3797	0
High pheromone with exploration	best	**12928**	**564**	36
	avg	**12967**	**546.8**	36
	std	44.3641	13.5536	0

Table 5.4: Experimental results for different configurations. The best results are shown in bold. Note that the best solution with respect to cost and number of performed tasks does not have the lowest standard deviation.

In Figure 5.2 the configuration of parameters for which MACS$^+$ delivers the best results can be seen in comparison with a Greedy and a Random approach. The Greedy approach consists of MACS$^+$ with $q_0 = 1.0$, which results in the algorithm always taking the connection with the lowest possible cost. This is pure exploitation without exploring good solutions. The Random algorithm, on the other hand, only chooses connections according to the biased random equation (see Equation 3.2 and 3.3) and only includes connection costs in the decision process to give better connections a higher probability of being chosen. As can be seen in Figure 5.2 the Random algorithm serves as a baseline. Furthermore, the Greedy algorithm arrives at a local optimum after only 7 iterations and does not improve further, while the MACS$^+$ algorithm does show improvement as time continues.

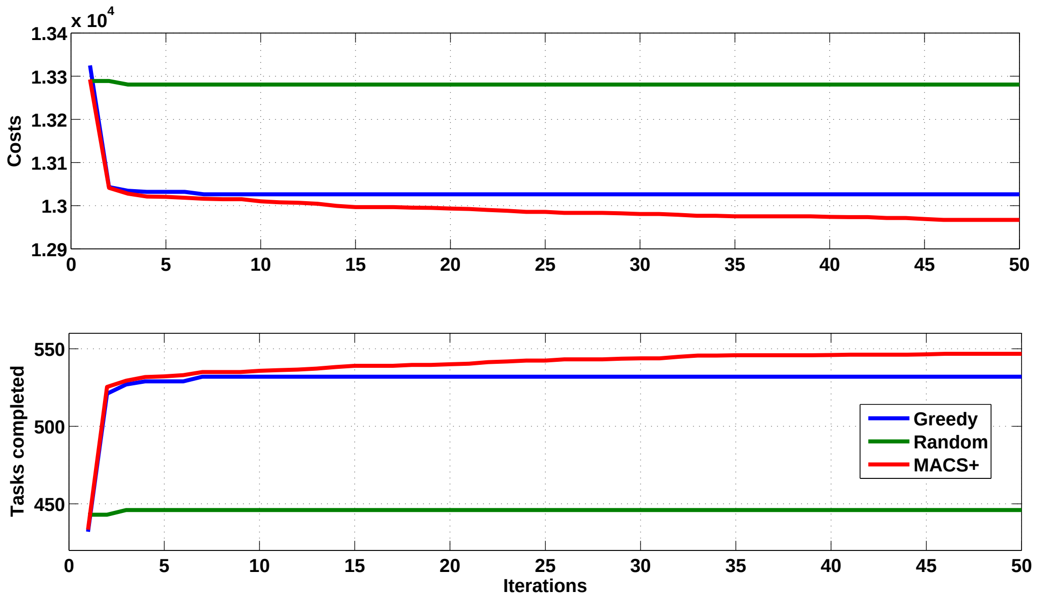

Figure 5.2: Experimental results; a Greedy and Random algorithm next to MACS.

5.4.2 Results with a lunch break

In Figure 5.3 and Table 5.5 the results for the experiments with the lunch break included in a shift can be seen. The total number of tasks performed has dropped compared to the experimental results where the lunch break was not included. Moreover, the solution quality increases when setting q_0 lower, thus increasing the probability for exploration instead of exploitation. The opposite was seen in the experiments without a break, where lowering the value of q_0 decreased the quality of the solution, because of too little exploitation of good moves.

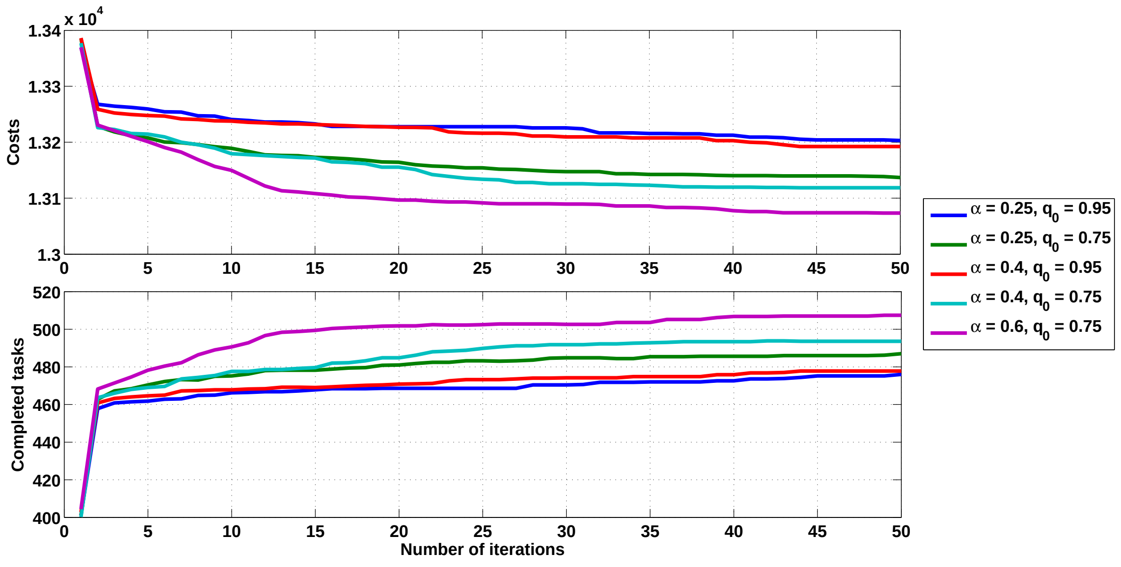

Figure 5.3: Experimental results for different configurations with a lunch break.

From Table 5.5 it can be seen that the algorithm performs best with $\alpha = 0.6$ and $q_0 = 0.75$, not only with respect to the total number of tasks completed and the lowest cost, but also with respect to the standard deviation when averaging the results over 5 runs.

		Costs	Tasks	Back
$\alpha = 0.25$, $q_0 = 0.95$	best	13128	498	36
	avg	13203	476	36
	std	60.4637	18.3984	0
$\alpha = 0.25$, $q_0 = 0.75$	best	13038	511	36
	avg	13137	487	36
	std	96.9719	26.0096	0
$\alpha = 0.4$, $q_0 = 0.95$	best	13104	499	36
	avg	13192	477.8	36
	std	71.2583	19.3830	0
$\alpha = 0.4$, $q_0 = 0.75$	best	**13036**	**516**	**36**
	avg	13119	493.6	36
	std	47.5235	13.0115	0
$\alpha = 0.6$, $q_0 = 0.75$	best	13050	511	36
	avg	**13073**	**507.4**	**36**
	std	**13.6931**	**2.3022**	**0**

Table 5.5: Experimental results for different configurations with a lunch break.

5.5 Discussion

To ensure that ants would return to their base station we implemented an urge to go back in Prototype Alpha (see section 3.3.2), that would increase as the end of a shift would come nearer. Results from the experiments with Prototype Alpha showed that not all ants were able to return in time before their shift ended. The replacement of this urge with a location based time indication of how much time will be needed to return to a base station from the current position ensures that the ants will choose a path toward their base station at the appropriate moment in their shift.

Although there was an increase in the number of ants that were able to return in time, there were still ants not arriving at a base station at the end of their shift. This was largely due to the fact that other ants had already travelled over the same shortest path which meant that those tasks were already completed and the next ant was unable to take that path home. To solve this problem we implemented a deadheading feature to the algorithm, which enables ants to travel as a passenger on a train (task) that has already been serviced by another ant. This resulted in a 100% successful return rate. Moreover, ants are now less likely to arrive in a sink node where no other tasks can be performed. To ensure that deadheading is kept to a minimum the number of deadheading tasks is subtracted from the number of actual performed tasks in the cost function (Equation 5.2). Thus, increasing the cost of a solution if more deadheading tasks are performed which make those solutions less likely to be chosen than the best solution found so far.

The increase in the number of ants that return to their base station through the hard constraint that the return path has to be chosen a the right point in time during a shift had an adverse effect on the overall number of executed tasks. A better local search was needed to ensure the chosen connections were not only cost effective on the short term but also cost effective for the entire solution on the long term. The look ahead function (see section 5.2.3) was introduced to perform a better local search by effectively looking ahead what the long term effects were of choosing a particular connection.

Finally the lunch break was implemented to be able to comply with union laws for break times during longer shifts [35] (i.e. more than 5:30 hours) and to be able to compare our solutions better with the NS Enschede dataset solution, which also incorporates a lunch break. Experiments conducted with the lunch break in place showed a decrease in the number of tasks performed, which was expected due to the fact that there is now less time to perform tasks.

With the four erroneous tasks removed from the NS Enschede dataset solution (see section 3.1 for details) there are still 580 tasks remaining compared to the 564 in our best solution, as can be seen in Table 5.4. However, the best solution is achieved without a lunch break. If the lunch break is included the number of tasks executed drops to 516 at best, as can be seen in Table 5.5.

Chapter 6

Discussion

6.1 Discussion

Prototype Alpha of MACS$^+$ gave insight in the relative impact of different parameters and showed that in order for all ants to return to their base station stricter rules were necessary than simply an urge to start the return path towards a base. Apart from that, the lunch break was not implemented to keep the scheduling problem from becoming too complex. Moreover, changes were made to the way in which the cost of taking a connection were calculated. The level of pheromone needed to have more influence in the connection choice, which was achieved by increasing the amount of default pheromone to 0.5.

With the introduction of deadheading (section 5.2.2) and the implementation of a hard cutoff point for ants to return to their base station (section 5.2.5) in Prototype Beta, every ant was able to end their shift at the station where they started. The introduction of deadheading means that ants are less likely to end their shift in a sink node that is not their base station, while the hard cutoff point enables ants to start the return journey to their base station at the appropriate time. Moreover, the introduction of a lunch break for shifts longer than 5:30 hours means that the crew schedules now adhere to Dutch union laws [2].

The implementation of the lunch break and a hard cutoff point to return to base had a negative effect on the number of tasks executed by each ant. Therefore, a better search heuristic was needed in the form of a look ahead function to achieve better local search results. The look ahead function enables ants to make local choices while taking into account the impact of that choice in a later part of the shift. The look ahead function can for example lead an ant away from a sink node well before it arrives there. Although the number of tasks performed per ant rose with the addition of the look ahead, the decrease in tasks executed compared to MACS$^+$ without the lunch break constraint was not resolved.

Results from the experiments with Prototype Beta without the lunch break are compared in Figure 5.2 with a greedy and a random search algorithm. The graph shows clearly that the random algorithm performs poorly, while the greedy algorithm increases solution quality in the first iterations only to get caught in a local optimum after which there are no more improvements. The MACS$^+$ approach shows the same sharp increase in solution quality in the first few iterations but in contrast to the greedy algorithm there is a steady decrease of costs and increase of tasks as time progresses.

A comparison with the solution provided in the NS Enschede dataset was performed. However, we found out that four tasks within the NS solution were not contained within the NS Enschede dataset. Those four tasks were replaced by idle time. The NS solution can be found in Appendix A. Even with the removal of four tasks the total number of tasks was 580 against 564(avg: 546.8) with the MACS$^+$ approach without the lunch break, and 511(avg: 507.4) including the lunch break (see Table 5.4 and 5.5). The best solutions found with MACS$^+$ with and without the lunch break constraint can be found in Appendix B.

6.2 Conclusion

The results from both Prototype Alpha as well as Prototype Beta enable us to answer the research questions stated in the introductory chapter (see 1.2).

How well does the MACS algorithm lend itself for a crew scheduling problem in a railway time table setting?

The MASC$^+$ algorithm does lend itself well to be used in a crew scheduling setting where there are no constraints on computation time. When comparing the solutions of MACS$^+$ with the solution provided in the NS Enschede dataset it can be seen that the results without the lunch break approach the quality of the NS. However, the inclusion of the lunch break has an adverse effect on the number of tasks executed and needs more research to increase the solution quality.

How well does the MACS algorithm lend itself for a crew re-scheduling problem in a railway disruption management setting?

In its current form MACS$^+$ will not be able to provide solutions fast enough to use it in a crew re-scheduling setting where time is of the essence. However, it is possible to use the best solution found so far after each iteration of the algorithm. This means that a choice can be made to run the algorithm for only a small number of iterations and use the best solution found at that point.

The current MACS$^+$ algorithm can be employed to handle the scheduling of railway crews when there is no time constraint such as in a disruption management setting. However, due to the characteristics of the algorithm there are valid solutions from two iterations onwards. One iteration needs on average four minutes to complete, which may decrease if the amount of exploration is greater. This does not hold for the first iteration in which only random choices are made. Therefore, after two iterations solutions are not optimal, but could be used in the event of an emergency.

The next and final section discusses the possible improvements that could be researched in order to not only improve the solution quality of MACS$^+$, but also achieve a reduction in computation time.

6.3 Future work

6.3.1 Employee start time

As mentioned in section 3.3.3 it is assumed that four ants beginning a shift every hour during the two rush hour periods in a day and one ant per hour in dull periods is sufficient to arrive at an optimal crew schedule (see figure 3.2 for the number of departures per hour on one day). However, the moment in time ants begin a shift can greatly influence the outcome of a crew schedule. Too many ants starting at one point in time means there will be too few tasks to execute and too few ants will results in too many tasks being left unaccomplished. More experiments need to be conducted in order to test the effect of varying the number of ants starting per hour and find the start times for an optimal crew schedule. Pheromone levels could be used to guide the choice of when to start shifts, as this mechanism is already implemented to guide the choice of which tasks to execute.

6.3.2 Parallelization

At the moment the entire system runs in one thread. The choosing of tasks is therefore done sequentially. For larger numbers of m ants the speedup cannot be guaranteed to be m times faster than the single threaded version due to overhead, specifically in waiting for shared memory. However, we feel that with efficient memory management a substantial drop in computation time could be achieved by parallelizing the MACS$^+$ algorithm.

Ellabib et. al. [23] mention that most parallel implementations (an overview of some implementations can be found in [18, 22]) can be divided in coarse-grained models and fine-grained models. In the fine-grained models the ants are divided in very small sub-populations that run on different processors and exchange small amounts of information frequently. This is suitable for a massively parallel architecture. The coarse-grained models consist of a few larger sub-populations where each populations runs on a different processor and information is exchanged at defined steps. This is suitable for clustered computers and the MACS$^+$ algorithm falls in this latter category, where all solutions are constructed before a global pheromone update is done for all colonies.

6.3.3 Collective intelligence

When the best solution is selected among the solutions created by the ants a global pheromone update will take place. This update only takes into account the links that are contained within the best solution. This means that all shifts created by the different ants are treated equally, even when one ant might only have completed 4 tasks while another might have done 18 or 20. It seems natural to change this. Ants that have made a considerable addition to the overall solution quality should be able to deposit larger amounts of pheromone on the connections they have chosen, while ants that did not contribute should deposit only small amounts.

Wolpert et al. [37] discuss Collective Intelligence (COIN), which is a large, sparsely connected neural network. The distinguishing feature of a COIN is that there is no centralized control, no centralized objective function, but rather the collective effects of individual agent each modifying their behaviour via their individual optimization algorithm. To achieve this in a MACS the central objective function should then be replaced by a individual optimization function for each ant.

6.3.4 Conductor & train driver

The minimal crew necessary to operate a train consists of a train driver and a conductor. For the experiments conducted with the MACS$^+$ algorithm we chose to schedule only one generic crew member per train. The solutions created by MACS$^+$ are only directly usable in the field if the generic crew member is seen as a duo of crew members where one is a train driver and the other is a conductor. Due to time constraints it was not possible to conduct additional experiments with the two types of crew member.

In order to work with two employee types the cost function should incorporate the amount of tasks that were performed by two different crew members, because one task can only be performed if one employee of each type executes that task. Moreover, the weights of connections should include a different cost for tasks that have no crew members and tasks that already have one crew member of another type present. Choosing the latter task will decrease the cost of a final solution as this task will be executed properly. Although adding the different types of crew members will constrain the crew scheduling problem further, it will results in proper solutions usable for scheduling individual employees in the field.

Appendix A

NS solution

Below is the solution from the NS Enschede dataset. The four tasks that were not contained within the NS Enschede dataset but were present in the solution are highlighted in red. These four tasks were seen as idle time when making the comparison to the solutions found with MACS$^+$. The number of tasks, including the four unknown tasks, is 584.

```
Crewbase,Duration,Costs,Tasks,
3,31140,45540,780,887,1048,1377,1893,2766,3397,3786,4041,4162,4592,4719,4866,5386,5801,5975,
3,32340,46740,6570,6741,6934,7335,8004,8397,8564,8730,8884,9034,9722,10164,10728,10969,11089,11135,
3,19740,34140,64,96,155,336,764,993,1253,1652,1890,2196,2436,2872,3148,3272,
3,31140,45540,4999,5088,5211,5452,5868,6057,6285,6638,6856,7157,7409,7908,8231,8377,8868,8987,9143,9609,9944,10071,
3,28740,43140,201,272,382,652,1132,13641,2309,2732,3024,3507,3695,4088,4271,4647,4891,5010,
3,31140,45540,6863,6975,7135,7460,7984,8216,8455,8782,8973,9213,9395,9727,9923,10014,10351,10427,10548,10892,11097,11143,
3,31740,46140,1135,1250,1421,1753,2257,2666,2887,3396,4147,4466,4610,4755,4895,5054,5369,5892,6237,6406,,
3,19740,34140,68,95,154,334,762,990,1252,1657,1897,2201,2438,2871,3146,3270,
3,29520,43920,7668,7855,8064,8449,8994,9574,9837,10231,10593,10698,10936,11009,11076,11180,11205,
3,28800,43200,333,451,612,1261,1810,2015,2478,2823,2991,3334,4213,4725,5032,5181,
3,19740,34140,6188,6289,6428,6724,7223,7454,7725,8121,8347,8620,8828,9217,9445,9546,
3,31140,45540,1144,1257,1424,1760,2263,2482,2712,3049,3246,3495,3693,4083,4333,4450,4884,4997,5154,5681,6114,6292,
3,31440,45840,830,1005,1211,1627,2278,2657,2828,3136,3253,3411,4345,5081,5592,5923,6075,
3,31200,45600,130,195,319,1022,1352,1534,1872,2161,2746,3109,3978,4563,4936,5185,5309,
3,25440,39840,5626,5773,5942,6281,6878,7280,7475,8053,8238,8426,8761,9277,9711,9845,
3,30540,44940,6518,6630,6780,7094,7594,8639,9549,10107,10532,10796,10945,11016,
3,31500,45900,6728,6869,7071,7741,8269,8456,8988,9079,9202,9433,9771,9913,10074,10291,10418,10605,10732,10972,11090,11136,
3,29460,43860,9,43,320,646,768,958,1866,3028,3945,4355,4518,
3,31140,45540,5585,5673,5811,6067,6512,6739,6980,7362,7605,7914,8172,8625,8887,9008,9431,9534,9666,10082,10377,10496,
3,24000,38400,21,36,236,542,1148,1580,1779,2127,2280,2461,3048,3480,3645,
3,31200,45600,7085,7240,7436,8119,8602,8776,9270,9351,9466,9679,9992,10530,10919,11103,11172,11189,
3,31200,45600,1370,1527,1737,2514,2804,2952,3234,3464,3965,4290,5131,5724,6123,6401,6542,
3,32340,46740,1528,1643,1811,2129,2595,3555,4319,5090,6024,6447,6753,6894,
3,32040,46440,5333,5479,5633,6315,6882,7796,8279,8704,8754,9334,9499,9733,9890,10177,10356,10435,
3,30540,44940,453,548,682,980,1506,1751,2013,2395,2606,2864,3081,3856,4172,4322,4471,4608,4775,18812,19175,19363,
3,32340,46740,192,265,370,640,1124,13641,2165,2317,3105,4203,4387,4648,4841,5224,5472,5596,
3,31800,46200,6392,6522,6712,7352,7884,8105,8551,8690,8847,9361,10047,10276,10646,10961,11046,
3,31200,45600,5765,5891,6046,6637,7143,7349,8000,8111,8271,8547,8969,9642,10113,10393,10575,10649,
3,31200,45600,6058,6186,6359,6989,7504,7721,8355,8454,8600,8864,9265,9436,9608,9856,9996,10176,10317,10603,10772,10839,
3,31200,45600,7611,7723,7896,8206,8672,8867,9077,9545,9584,9847,10089,10255,10456,10705,10826,10978,11070,11187,11216,11222,
3,31200,45600,3734,3845,4002,4631,4872,5365,6068,6513,6740,6981,7360,7604,7913,8171,8623,8886,9007,
3,24600,39000,5170,5287,5436,5987,6435,6631,7221,21931,7836,7995,8191,8788,9196,9347,
3,31140,45540,457,551,688,991,1514,1750,2011,2383,2594,2863,3080,3484,3739,3859,4307,4422,4579,5093,5502,5671,
3,30540,44940,2612,2710,2854,3137,3552,3887,4066,4440,4491,4799,5099,5339,5616,5986,6179,6453,6690,7156,7474,7637,
3,31200,45600,2478,2610,2785,3352,3767,3932,4430,4516,4641,4883,5279,5454,5667,5983,6174,6446,6685,7148,7468,7639,
3,32340,46740,775,880,1041,1369,1894,2767,3548,4520,5226,5419,5818,6081,6212,
```

Appendix B

MACS$^+$ solutions

B.1 MACS$^+$ solution without a lunch break

Below the MACS$^+$ solution without the lunch break constraint can be seen. Here the tasks highlighted in red are the deadheading tasks where ants travel as passengers. These tasks are included in the cost of the solution as idle time.

```
Crewbase,Duration,Costs,Tasks
3,31260,45660,(9),43,69,120,564,1046,1254,1923,2259,2636,2945,3038,3179,3918,4435,4753,4900,
3,28860,43260,9,(43),(69),374,560,988,1139,1341,2023,2524,2727,3136,3253,3411,3945,4355,4518,
3,31140,45590,68,95,154,334,762,990,1252,1657,1897,2201,2438,2871,3146,3270,3600,3745,3895,4213,4725,5032,5181,
3,28020,42420,64,96,155,336,764,993,1253,1652,1890,2196,2436,2872,3148,3272,3544,(4022),(4143),4490,
3,31200,45600,130,195,319,891,1420,1640,2127,2280,2461,3048,3480,3645,4022,4143,4292,4808,5218,5380,
3,31740,46140,192,265,370,640,1124,2165,2317,2582,2784,(4238),4416,4648,4841,5224,5472,5596,
3,31740,46140,201,272,382,652,1132,2167,2458,2822,3186,4474,(4841),(5224),(5472),(5596),
3,25620,40020,333,451,612,1261,1810,2015,(2478),(2610),3108,3647,4061,4225,4592,4719,
3,23160,37560,453,548,682,980,1506,1751,2013,2395,2606,2864,3081,3484,3739,3859,4145,4230,4358,
3,26760,41160,457,551,688,991,1514,1750,2011,2383,2594,2863,3080,3716,4220,4369,4914,
3,31140,45540,780,887,1048,1627,2278,2657,2828,2986,3144,3304,3629,4147,4466,4610,4755,4895,5054,(5592),(5923),(6075),
3,30840,45240,830,1005,1211,1912,2307,2490,2664,2823,2991,3334,3856,4172,4322,4471,4608,4775,5081,5592,5923,6075,
3,30240,44740,775,880,1041,1369,1894,2767,3396,4076,4631,5301,5619,5771,5929,
3,31740,46140,1135,1250,1421,1753,2257,2480,2711,3049,3246,3495,3693,4083,4333,4450,4703,(5170),(5287),5801,5975,6542,
3,30540,44940,1144,1257,1424,1760,2263,2482,2712,3465,3599,3829,4498,4676,4935,5114,5681,6114,6292,
3,26880,41280,1370,1527,1737,2388,2860,3037,3433,3564,3706,4231,4642,4804,(5170),(5287),5633,
3,24660,39060,1528,1643,1811,2129,2595,3516,3952,4366,4688,5409,
3,30600,45050,2478,2610,2785,3352,3767,3932,4307,4422,4579,5508,6031,6690,7156,7474,7637,
3,31740,46140,2612,2710,2854,3137,3552,3728,3933,4238,(4416),(4648),4985,6342,7048,7530,7851,8015,
3,31200,45600,3734,3845,4002,4525,4919,5083,5460,5584,5738,6302,6778,6979,7447,7607,7807,8462,8904,9074,
3,28020,42420,4999,5088,5211,5466,5874,6057,6285,6638,6856,7157,7409,7908,8231,8377,8674,(9160),(9271),9574,
3,28800,43200,5170,5287,5436,5987,6435,6631,(7085),(7240),7688,8096,8691,9025,9174,9431,9534,(9923),(10014),
3,27720,42120,5286,5375,5499,5964,6530,6911,7102,7286,7471,8036,8223,8404,(8565),9034,9338,9782,
3,31440,45840,5626,5773,5942,6281,6878,7280,8111,8271,8547,8969,9642,10113,10393,10575,10649,
3,25620,40020,5585,5673,5811,6067,6512,6739,6980,7362,7605,7914,8172,8625,8887,9008,9270,9351,9466,9679,
3,31200,45600,5765,5891,6046,6637,7143,7349,7836,7995,8191,8788,9196,9347,9685,9784,9903,10296,10593,10698,
3,30000,44400,6058,6186,6359,6989,7504,7721,8208,8565,8739,9067,9728,10082,10377,10496,10765,
3,29940,44340,6188,6289,6428,6724,7223,7454,7725,8121,8347,8620,8828,9217,9445,9546,9769,(10135),(10427),10772,10839,
3,29640,44040,6392,6522,6712,7352,7884,8105,8551,8690,8847,9361,9711,9845,10135,10222,10334,10703,
3,25320,39720,6570,6741,6934,7335,8004,8397,8564,8730,8884,(9463),(9603),10004,10075,10171,10354,
3,30540,44940,6518,6630,6780,7094,7594,8639,9092,9475,9919,10110,10278,10603,10961,11046,
3,29640,44040,6728,6869,7071,7741,8269,8456,8868,8987,9143,9609,9944,10071,10351,10427,10548,10892,
3,25620,40020,6863,6975,7135,7460,7984,8216,8455,8782,8973,9213,9395,9727,9923,10014,10226,10288,10375,10557,
3,27000,41400,7085,7240,7436,8119,8602,8776,9160,9271,9418,9855,10167,10284,10559,10641,10945,11016,
3,30300,44700,7668,7855,8064,8449,8994,9463,9603,9916,10003,10125,10507,10791,10887,11094,11124,11222,
3,28140,42540,7611,7723,7896,8206,8672,8867,9077,9360,9523,9732,9889,10342,10432,10730,10880,11103,11172,11189,
Tasks: 564 DeadHeadingTasks: 32 Best: 12927.777777777777 Antnr: 22 Back: 36
```

B.2 MACS$^+$ solution with a lunch break

Below the MACS$^+$ solution with the lunch break constraint can be seen. As in the MACS$^+$ solution without the lunch break, the tasks highlighted in red are the deadheading tasks where ants travel as passengers. These tasks are included in the cost of the solution as idle time.

```
Crewbase,Duration,Costs,Tasks
3,29880,44280,(9),(43),69,120,564,1046,1254,1962,2147,2332,2701,3260,3594,3752,3890,4034,(4703),
3,29880,44280,9,43,(69),374,560,988,1139,1341,2443,2885,3495,3693,4083,4333,4450,4703,
3,31140,45540,64,96,155,336,764,993,1253,1652,2297,2656,3158,3363,3597,3944,4725,5032,5181,
3,31740,46140,68,95,154,334,762,990,1252,1657,2373,3166,3294,4565,4936,5185,5309,
3,31200,45600,130,195,319,891,1420,1640,2127,2636,2945,3038,3179,3434,4259,4808,5218,5380,
3,25740,40140,201,272,382,652,1132,2167,2786,3142,4237,
3,24720,39120,192,265,370,640,1124,2165,3113,3582,4099,4255,
3,30600,45000,333,451,612,1449,2362,3447,3835,4020,4416,4648,4841,5224,5472,5596,
3,31140,45540,457,551,688,991,1514,1750,2011,2383,3031,3799,4086,4270,4647,4891,5010,5333,5771,
3,28800,43200,453,548,682,980,1506,1751,2013,2395,3032,3507,3695,(4086),(4270),(4647),(4891),(5010),(5286),(5375),
3,31740,46140,780,887,1048,1377,1893,2766,3857,4639,5419,5818,6081,6212,
3,27720,42120,830,1005,1211,1627,2278,2657,2828,3300,3448,3612,3918,4435,4753,4900,5040,5183,5343,
3,31140,45640,775,880,1606,2003,2618,3304,3629,4147,4466,4610,4755,4895,5054,5592,5923,6075,
3,26940,41340,1135,1250,1421,1753,2257,2480,2711,3049,3644,3948,4135,4371,4546,5093,5502,5671,
3,29940,44340,1144,1257,1424,1760,2263,2482,2712,3463,4754,4940,(5419),(5818),(6081),(6212),
3,28800,43200,1370,1527,1737,2388,2860,3037,3433,3859,4145,4230,4358,4601,(5419),(5818),(6081),(6212),
3,31740,46140,1528,1643,1811,2129,2595,3516,4294,4965,5823,6024,6447,6753,6894,
3,25620,40020,2478,2610,2785,3352,3767,3932,4307,4719,4866,5386,5801,(5975),(6392),(6522),
3,29340,43740,2612,2710,2854,3137,3552,3728,3933,4238,4803,5095,5281,5517,5702,6302,6778,6979,7447,
3,30000,44400,3734,3845,4002,4525,4919,5083,5460,5975,(6392),(6894),7286,7471,7688,8364,8719,8880,
3,29940,44340,4999,5088,5211,5466,5874,6057,6285,6638,7271,7664,8108,8387,8973,9213,9395,9727,9923,10014,
3,28200,42600,5170,5287,5436,5987,6435,6631,7221,(7721),8208,8356,8535,9084,(9463),9603,9916,
3,30420,44820,5286,5375,5499,6067,6512,6930,7983,8568,9334,9499,9732,9889,10185,
3,31440,45840,5626,5773,5942,6281,6878,7280,7475,8053,8238,8426,8761,9277,9711,9845,(10135),(10222),10575,10649,
3,31740,46140,5585,5673,5811,(6067),(6512),6739,6980,7362,8024,8391,8892,9093,9310,9651,9952,10150,10239,10429,(10649),
3,31200,45600,5765,5891,6046,6637,7143,7349,7836,8377,8674,8987,9143,9609,9944,10071,10351,10698,
3,29280,43680,6058,6186,6359,6989,7504,7721,8404,8884,9034,9338,9782,10171,10603,
3,29940,44340,6188,6289,6428,6724,7223,7454,7725,8121,8721,9165,9556,9830,9974,10176,10317,(10603),10772,10839,
3,20880,35280,6392,6522,6712,7352,7884,8105,8551,9008,9270,9351,9466,9679,
3,26460,40860,6570,6741,6934,7335,8004,8397,8564,9160,9271,9418,9855,10167,10284,10559,10641,
3,29940,44340,6518,6630,6780,7094,7594,8639,9407,10107,10532,10796,10945,11016,
3,29280,43680,6728,6869,7071,7741,8269,8690,9175,9287,9531,9604,9714,9911,10216,10598,10728,10969,
3,31740,46140,6863,7607,7807,8909,9455,9672,9786,10294,10562,10730,10880,11103,11172,11189,
3,27600,42000,7085,7240,7436,8119,8602,8776,(9160),9546,9769,10135,10222,10334,10703,10961,11046,
3,30300,44700,7668,7855,8064,8449,8994,9463,10003,10125,10507,10791,10887,11094,11124,(11222),
3,30600,45000,7611,7723,7896,8206,8672,8867,9077,9360,9843,10395,10705,10826,10978,11070,11187,11216,11222,
Tasks: 511 DeadHeadingTasks: 34 Break: 36 Best: 13077.777777777777 Antnr: 26 Back: 36
```

Bibliography

[1] Eclipse 3.5.2. `http://www.eclipse.org/`. Accessed: 2011.

[2] Abbink, E., Fischetti, M., Kroon, L., Timmer, G., and Vromans, M. Reinventing crew scheduling at Netherlands railways. *INTERFACES 35*, 5 (2005), 393–401.

[3] Abbink, E., Mobach, D., Fioole, P., Kroon, L., vander Heijden, E., and Wijngaards, N. Real-time train driver rescheduling by actor-agent techniques. *Public Transport* (2010), 1–20.

[4] Abbink, E. J. W., Mobach, D. G. A., Fioole, P. J., Kroon, L. G., van der Heijden, E. H. T., and Wijngaards, N. J. E. Actor-agent application for train driver rescheduling. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1* (Richland, SC, 2009), AAMAS '09, International Foundation for Autonomous Agents and Multiagent Systems, pp. 513–520.

[5] Braysy, O., and Gendreau, M. Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *TRANSPORTATION SCIENCE 39*, 1 (2005), 104–118.

[6] Bussieck, M. R., Winter, T., and Zimmermann, U. T. Discrete optimization in public rail transport. *Math. Programming 79* (1997), 415–444.

[7] Caprara, A., Monaci, M., and Toth, P. A global method for crew planning in railway applications. In *Vo, S., Daduna, J.R. (Eds.), Computer-Aided Scheduling of Public Transport, Lecture Notes in Economics and Mathematical Systems, 505* (2001), Springer.

[8] Castro, A., and Oliveira, E. Using specialized agents in a distributed MAS to solve airline operations problems: a case study. In *International Conference on Intelligent Agent Technology* (2007), IEEE, pp. 473–6.

[9] Chaslot, G., Winands, M., van den Herik, J. H., Uiterwijk, J., and Bouzy, B. Progressive strategies for Monte-Carlo tree search. In *Joint Conference on Information Sciences, Salt Lake City 2007, Heuristic Search and Computer Game Playing Session* (2007).

[10] Chih-Chung, L., and Guang-Feng, D. Using ant colony optimization algorithm to solve airline crew scheduling problems. *International Conference on Natural Computation 4* (2007), 797–804.

[11] CLAUSEN, J., LARSEN, A., LARSEN, J., AND REZANOVA, N. J. Disruption management in the airline industry–concepts, models and methods. *Computers & Operations Research 37*, 5 (2010), 809 – 821.

[12] CRAWFORD, B., CASTRO, C., AND MONFROY, E. A hybrid ant algorithm for the airline crew pairing problem. In *MICAI 2006: Advances in Artificial Intelligence*, A. Gelbukh and C. Reyes-Garcia, Eds., vol. 4293 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 381–391.

[13] DE JONG, J., AND WIERING, M. Multiple ant colony systems for the busstop allocation problem. In *Proc. of the Thirteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'01)* (2001), pp. 141–148.

[14] DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematik 1* (1959), 269–271.

[15] DORIGO, M., BIRATTARI, M., AND STTZLE, T. Ant colony optimization – artificial ants as a computational intelligence technique. *IEEE COMPUT. INTELL. MAG 1* (2006), 28–39.

[16] DORIGO, M., AND BLUM, C. Ant colony optimization theory: a survey. *Theor. Comput. Sci. 344* (November 2005), 243–278.

[17] DORIGO, M., CARO, G. D., AND GAMBARDELLA, L. M. Ant Algorithms for Discrete Optimization. *Artificial Life 5*, 2 (1999), 137–172.

[18] DORIGO, M., AND DI CARO, G. *The ant colony optimization meta-heuristic*. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999, pp. 11–32.

[19] DORIGO, M., AND GAMBARDELLA, L. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on 1*, 1 (apr 1997), 53 –66.

[20] DORIGO, M., AND KRZYSZTOF, S. An Introduction to Ant Colony Optimization. *IRIDIA Technical Report Series* (2006).

[21] DORIGO, M., MANIEZZO, V., AND COLORNI, A. Ant system: Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS PART B-CYBERNETICS 26*, 1 (FEB 1996), 29–41.

[22] DORIGO, M., AND STÜTZLE, T. *Ant Colony Optimization (Bradford Books)*. The MIT Press, July 2004.

[23] ELLABIB, I., CALAMAI, P., AND BASIR, O. Exchange strategies for multiple ant colony system. *Inf. Sci. 177* (March 2007), 1248–1264.

[24] FORES, S., PROLL, L., AND WREN, A. TRACS II: a hybrid IP/heuristic driver scheduling system for public transport. *JOURNAL OF THE OPERATIONAL RESEARCH SOCIETY 53*, 10 (2002), 1093–1100.

[25] FORSYTH, P., AND WREN, A. An ant system for bus driver scheduling. In *The 7th International Workshop on Computer-Aided Scheduling of Public Transport* (1997).

[26] FRELING, R., WAGELMANS, A., AND PAIXAO, J. An overview of models and techniques for integrating vehicle and crew scheduling. In *Lecture notes in economics and mathematical systems*, vol. 471. Springer-verlag Berlin, 1999, pp. 441–460.

[27] GAGNÉ, C., GRAVEL, M., AND PRICE, W. L. A look-ahead addition to the ant colony optimization metaheuristic and its application to an industrial scheduling problem, 2001.

[28] GAJPAL, Y., AND ABAD, P. Multi-ant colony system (macs) for a vehicle routing problem with backhauls. *European Journal of Operational Research 196*, 1 (2009), 102 – 117.

[29] GAMBARDELLA, L. M., TAILLARD, E., AND AGAZZI, G. Macs-vrptw: A multiple ant colony system for vehicle routing problems with time windows. In *New Ideas in Optimization* (1999), McGraw-Hill, pp. 63–76.

[30] HUISMAN, D. A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research 180*, 1 (2007), 163 – 173.

[31] MAO, X., MORS, A., ROOS, N., AND WITTEVEEN, C. Coordinating competitive agents in dynamic airport resource scheduling. In *Proceedings of the 5th German conference on Multiagent System Technologies* (2007), MATES '07, Springer-Verlag, pp. 133–144.

[32] MICHEL, R., AND MIDDENDORF, M. An island model based ant system with lookahead for the shortest supersequence problem. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature* (London, UK, 1998), PPSN V, Springer-Verlag, pp. 692–701.

[33] MONTEMANNI, R., GAMBARDELLA, L., RIZZOLI, A., AND DONATI, A. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization 10* (2005), 327–343.

[34] SHIBGHATULLAH, A., ELDABI, T., AND RZEVSKI, G. A framework for crew scheduling management system using multi-agents system. In *ITI 2006 Proceedings of the 28th International Conference on Information Technology Interfaces* (Zagreb, Croatia, 2006), V. Luzar and V. Hiluz, Eds., University of Zagreb, pp. 379–84.

[35] VEELENTURF, L. P., POTTHOFF, D., HUISMAN, D., AND KROON, L. G. Railway crew rescheduling with retiming. *Transportation Research Part C: Emerging Technologies* (2010).

[36] WALKER, C. G., SNOWDON, J. N., AND RYAN, D. M. Simultaneous disruption recovery of a train timetable and crew roster in real time. *Comput. Oper. Res. 32* (2005), 2077–2094.

[37] WOLPERT, D. H., TUMER, K., AND FRANK, J. Using collective intelligence to route internet traffic. In *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS* (1999), MIT Press, pp. 952–958.

[38] WREN, A., AND ROUSSEAU, J.-M. Bus driver scheduling - an overview, 1993.