UNIVERSITY OF TWENTE.

Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)
Department of Discrete Mathematics and Mathematical Programming (DMMP)

Master thesis

Workforce scheduling algorithms at Grolsch Brewery Enschede

Selmar van der Veen
October 2015

Assessment committee

Prof.dr. M.J. Uetz
Ing. D. Temmink-Breden
Dr.ir. G.F. Post
Dr. J.C.W. van Ommeren

Preface

This report gives the overview of the research performed at the department of Quality Control at Grolsch Brewery Enschede. The focus of this report will be on the final project phase of the research, which started as an internship. The final results of the internship were presented in May 2015.

First of all, I thank Marc Uetz for being my supervisor the last year during my internship and final project. During the whole process and the writing of my thesis, all comments were very useful.

Starting with the project at Grolsch was the first contact for me with industry. Thanks to the colleagues for this introduction and furthermore providing me with all the necessary information needed for my research. Especially those at the department of Quality Control and in particular Daniëlle Temmink who was my direct supervisor. All the information and feedback given, helped me a lot during my research. Last, my thanks are going to Karsten Herr, my first supervisor at Grolsch who found a new job at Thales during my research, for providing me with this assignment.

Selmar van der Veen

Abstract

In this thesis, a work force scheduling problem at Grolsch Brewery is examined. A number of tasks must be scheduled over a period of days with a certain number of analysts available to obtain an optimal schedule that results in a maximum number of free analysts available each period. Every task and analyst has its own properties, and there are other side constraints that all have to be taken into account. Therefore, as a first do-ability study, an integer linear program is formulated and solved with CPLEX as general purpose integer linear programming solver. This turned out to yield good solutions, yet be very time expensive. Moreover, it turned out that cost considerations prohibit the use of commercial solvers. Therefore alternative, heuristic scheduling algorithms have been implemented for the problem. A simulated annealing approach is used for this. A number of variants have been tested, for example different cooling schedules and different initial temperatures. An extensive computational study of those variations leads to an overview of the results that can be expected to be obtained for the practical deployment for the solution of this specific problem.

Contents

Chapter 1

Introduction

One of the main conditions for being a succesfully company is delivering good products. Within the Grolsch Brewery the department Quality Control is mainly responsible for monitoring the quality of mostly all products that are produced. They perform tests on all different products themselves but also on the machinery that is used during the production and packaging of their products. To perform all these tests about ten to fifteen analysts are working in the laboratory of Quality Control.

The Quality Control lab has traditionally been organized by workplaces. However, the management has observed that the time of staff for performing additional trainings or projects was fragmented, leading to them being postponed more than necessary. The question was if better scheduling of the workforce would give more control over the actual work distribution, and allow to build into the schedule also such additional tasks.

Chapter 2 describes the problem description. It starts with an overview of the activities that have to be performed by the department. This is followed by a description of the current situation at the laboratory of the department followed by the problem description of this research which is sketched by an example.

Chapter 3 gives an overview of related literature about similar topics.

To model the the problem, an integer linear programming model (ILP) is formulated and presented in Chapter 4 together with the objective function. The last section of the chapter proves NP-hardness of the problem also in combination with the selected objective function.

Because of NP-hardness of the problem and the fact that using an expensive commercial solver turned out to be out of the scope of the project, Chapter 5 presents a scheduling algorithm used for this specific problem. This local search algorithm is called simulated annealing and is explained in Section 5.3.

Chapter 6 gives an overview of the data that has been used. The computational results obtained with CPLEX are presented in Chapter 7. This chapter gives also the objective values for the datasets that are used to analyze the performance of the simulated annealing algorithm. The results of the performance analysis is presented in Chapter 8. Our computational results show that, with an alternative way of scheduling tasks at the Quality Control department, specifically using a simple simulated annealing algorithm, the main goal of unfragmented time for projects can indeed be achieved.

Chapter 2

Problem description

First an overview is given of all the work that has to be performed by the analysts at the department of Quality Control. A summary of all the different tasks will be presented and the current situation of distributing the tasks over the analysts will be explained by an example.

The specific instance used for the example gives in a simplified way the current work load situation and the desirable situation.

2.1 Overview tasks

To give an idea of the workload for the department Quality Control, an overview is given of the different types of tasks performed by the analysts working at the laboratory. The types will be explained in this section and are given below:

- daily tasks;
- periodic tasks;
- releases (called 'vrijgaven' in Dutch);
- projects.

First, it has to be mentioned that in most cases a task consists of a number of tests. For example, a test could be to determine the turbidity of a single sample. On a daily basis this test is executed multiple times on the same machine. The collection of all these tests is seen as one task.

Daily tasks are tasks with a release date that is equal to the due date and are performed on a daily basis. Examples of daily tasks are determining the turbidity of a sample or determining the alcohol percentage of a product.

Periodic tasks are performed on a less then daily basis. This means for example tasks that are processed by analysts once a week or once a month, therefore the due date is more then one day behind the release date. An example of a periodic task is the taking of airsamples in the packaging area of the brewery.

Another very important type is a release. If a product is ready for packaging, the department of Quality Control has to give permission over all the parameters linked to the specific product. If the parameters are within the predefined quality interval, a product will be packaged. Such a release has to be done within two hours after the sample enters the laboratory.

Within the two hours all tests have to be performed, but the distribution of the samples over the working places takes at most ten minutes.

When there is reffered to regular tasks, daily tasks, periodic tasks and releases are meant.

Projects are the last type of tasks. Analysts are working on projects to improve the quality of their department or other parts of the brewery. In the standard situation, the focus will be first on regular tasks and if there is time left, analysts will be scheduled to work on projects.

2.2 Current workload situation

In the current situation the lab is working with a fixed number of workplaces each day. Each of the workplaces is assigned a list of regular tasks. All the tasks vary in processing time due to the different amounts in test for a task each day. The length of each process at the brewery, the number of changes between processes and the type of process all have influence on the number of analysis and therefore the length of a task. Sometimes tasks are manually replaced to another analyst, because of the fact that the availability constraint could be violated.

Next to all the analysts that work on weekdays, almost every weekly evening and weekend an analyst is working at the lab. The presence of this analyst is necessary for the releases and next to the releases the analyst performs some other tests. All the evening and weekend tasks are not taken into account in the model. This is because evening and weekend tasks are tasks that have to be finished in that shift. For example, soms tasks in the evening have to be finished because they started earlier that day. Another situation that occurs in practice is that an analyst has to wait for a release, so he can better work on tasks than doing nothing.

Except all regular tasks that are performed by the analysts, the analysts are working on a lot of different projects throughout the year. These projects are performed to result in improvements which can be implemented at the department of Quality Control. In the current situation analysts are scheduled a day off for regular task to work on projects, which happens in advance. It is possible that an analyst that is scheduled on projects is performing regular tasks though, because of a high workload of other analysts.

Working on projects has not the highest priority at the laboratory. However, finishing projects earlier means a better performance of the department in the long run and a faster improvement due to the results of the projects.

2.3 Problem description

The scheduling of the projects reveals the real problem description of this research. To ensure that projects are completed in an as short as possible period, it is necessary to schedule analysts on the specific projects on a regularly basis. This has to be done in a way that the number of analysts that is available for regular tasks at the laboratory is satisfied.

Tasks can be assigned to the analysts under some restrictions. The goal of this research is to investigate the different ways to schedule regular tasks in such a way that the available time for projects is scheduled in an optimal way. This is done with the number of analysts available which means that it is not the goal to minimize the number of analysts working at the lab needed to perform all regular tasks.

For a project, it is better to cluster all available time in as few as possible days than to spread it over a lot of days. The last idea will for example lead to four days with on each one hour available for projects. It will be a better schedule to cluster these four hours on one day and let the same analysts work on the project that half day.

This leads to preferred schedules where the number of 'free' analysts is maximized. Furthermore, over all the analysts with work load, the work load has to be divided in such a way that the spare time is clustered as most as possible.

To illustrate this example with respect to the practical situation, Figure 2.1 shows two different schedules of a set of tasks for the same day. The diagram gives the total workload for four analysts on that day. The 'max' represents the total availability of each analyst. This maximum capacity is in most of the situations the same for each analyst on each day.

The left diagram can be seen as a representation of the current scheduling method, where analysts (*1*, *2*, *3*, *4*) are connected to a specific workplace (A, B, C, D) with associated tasks. Due to the variations in the processing times of tasks each day, the time that is not scheduled varies over the analysts. The shaded part of the diagram depicts the spare time of an analyst.

If another scheduling method is used for the situation in Figure 2.1, it is possible to create a schedule that is depicted at the right. Note that all the spare time is scheduled at the first analyst.

In Section 2.4 the idea illustrated in Figure 2.1 will be explained more accurately with an example set of analysts and tasks.

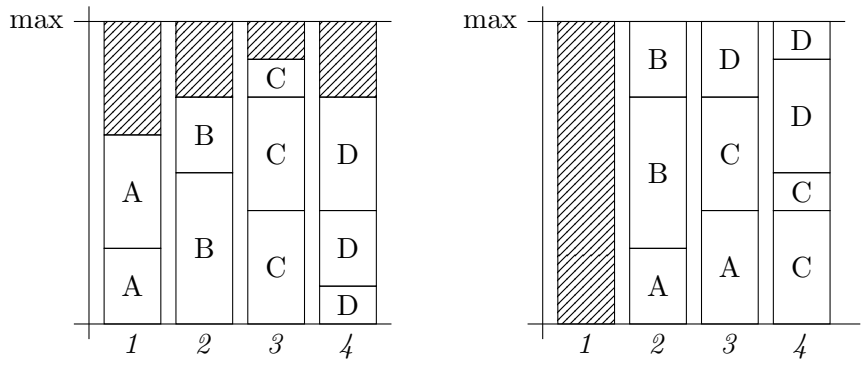


Figure 2.1: Illustrated situation with workplaces (left) and with a distribution of tasks over all available analysts (right).

2.4 Illustrative example

In Tables 2.1–2.3 all the data can be found that is used in this example. In total, the example consists of eleven tasks which have to be progressed by three analysts during a period of two days.

The purpose of the schedule is that each task is assigned to a specific analyst and day so that all the restrictions are taken into account.

It is of course not possible that the daily workload assigned to an analyst is more than the available time for the analyst. The available time for an analyst on a specific day can be found in Table 2.1. Furthermore, there may be restrictions on the interval in which a

		Day	
		1	*2*
Analyst	*1*	8	8
	2	8	8
	3	0	8

Table 2.1: Availability of analysts.

Task	Duration	Release date	Due date
1,2,4	2	1	1
3,6	2	1	2
5	4	1	1
7,8,9	4	1	2
10,11	2	2	2

Table 2.2: Data for each task.

		Task										
		1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
Analyst	*1*	1	1	1	0	0	0	1	1	0	0	0
	2	0	0	0	1	1	1	0	1	0	1	1
	3	0	0	0	0	0	0	0	0	1	1	1

Table 2.3: Qualifications for each analyst.

task may be progressed. This means that each task has a release date and a due date. The planning horizon for this example is only two days, which means that some of the tasks may be executed only on one of the two days while others can be scheduled both of the days. This data can be found in Table 2.2. At last, an analyst must have the qualification to perform a task. This is denoted by a '1' in Table 2.3.

It is possible that the constraints leave only one possibility for scheduling a certain task. For example, it is possible that a task may only be processed by a single analyst or on a specific day. In the example, this is true for task *9*. Only analyst *3* is allowed to process task *9*. Furthermore, analyst *3* is only available on the second day. This automatically means that task *9* will be scheduled on day *2*.

Two different schedules are given in Figure 2.2. Both situations are illustrative for the practical situation. Situation A is a schedule which is the consequence of scheduling according to the workplaces, where each task is assigned to a workplace. It is easy to see that there are small shaded parts for each analyst on the different days, which means there is time left over that day. Situation B gives a schedule where the tasks are optimally clustered over the available analysts, because tasks are not restricted to workplaces. This way of scheduling results in one analyst that is scheduled free for the second day and the time of other analysts is scheduled in a better way to process regular tasks.

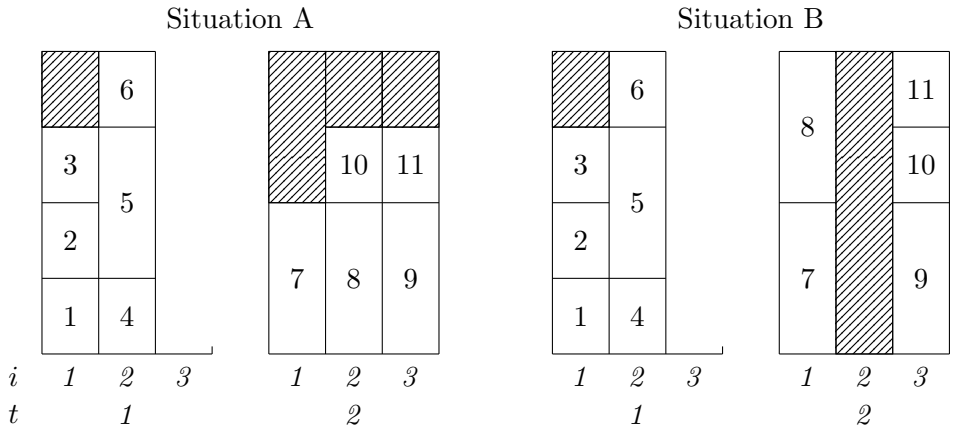


Figure 2.2: Current situation for the example dataset (A) and the situation where as few as possible analysts are used for scheduling all tasks (B). Analysts are represented by i and days by t.

Chapter 3

Related literature

Scheduling is a widely studied field in Operations Research. An extensive overview on scheduling topics is presented by Leung [7]. It contains definitions of the most common problems in practice as well as the classical scheduling problems. Job-shop scheduling, real-time scheduling and stochastic scheduling are all reviewed.

In Brucker et al. [2] more complex scheduling problems are studied. The focus is on resource-constrained project scheduling and complex job-shop scheduling.

Three different approaches are available to solve the wide variation of scheduling problems. The first one is to formulate the problem as an integer linear problem (ILP). Solution techniques for ILP's are a research area where a lot of research has been performed. Well-developed solvers are available to obtain solutions for an ILP. However, a lot of those problems formulated as an ILP are called NP-hard and therefore general commercial solvers are not able to find solutions in polynomial time. On the other hand, the second option to solve scheduling problems are approximation algorithms that find approximate solutions to optimization problems.

The last method to solve scheduling problems are local search algorithms. A number of them are presented in Aarts et al. [1]. Local search algorithms are searching for a solution with respect to a certain objective function when considering a number of candidate solutions, also called neighborhoods. Moving from those candidate solutions to another candidate solution the objective function is optimized and the algorithm terminates when certain stopping criteria are satisfied such as a maximum number of iterations or running time.

A number of common aspects of scheduling problems appear in the problem of this research. First of all, assigning tasks with a certain processing time to a number of analysts has similarities with the bin packing problem where a number of items must be assigned to a number of bins. The availabilities of the analysts are similar to the sizes of the bins.

However, extra constraints are imposed that prohibits assigning tasks to some analysts because each analyst has his own qualifications. Also some combinations of tasks may not be scheduled at the same analyst.

There are also a lot of problems in scheduling that are dealing with release and due dates, so is this problem. The processing of a task can only start when a task is released and has to be finished before the release date, otherwise extra costs are incurred.

Chapter 4

Integer linear programming

This section of the report deals with the formulation of the mathematical model. We start with an explanation of the input and a description of the data that is necessary. Also the output format is explained.

The information described before is used to formulate an integer linear programming model (ILP). This also includes an explanation of the variables used and the restrictions that are taken into account. The objective function will be formulated as well as some other possible objective functions are discussed.

4.1 Functionalities of the model

The input of the model consists of a parameter T which indicates the length of the planning horizon in days. A list of tasks is available with a release date, a due date and a processing time for each task. The release and due date are defining the interval in which the task has to be processed.

Some tasks may not be scheduled on the same day at the same analyst, due to time constraints. An example for this is two tasks with a processing time of more then two hours which have to be performed in the morning.

Furthermore a list of analysts is available containing all the information about them. It is known for which tasks they have the qualification and how many hours they are available each day.

It was decided to allocate the tasks to a day and an analyst. This means that each analyst gets a package of tasks for each day, scheduled in advance and based on the data and not the estimation of analysts. This gives the analyst the freedom to create his own schedule and gives the analyst enough total workload to be busy the whole day.

The model doesn't take into account the variability of processing times of a task due to the fact that not every analyst has the same experience for the tasks. This influence will be neglected because if an analyst performs a task more often, the processing times will be almost equal over all the analysts. Besides that, 'slow' en 'fast' tasks scheduled on the same day for an analyst will cancel out with high probability.

4.2 ILP-formulation

First the tasks, analysts and days are given. The total number of these parameters are respectively denoted by n, m and T. The associated indices can be found in Table 4.1.

Tasks	$j, k = 1, \ldots, n$
Analysts	$i, h = 1, \ldots, m$
Days	$t = 1, \ldots, T$

Table 4.1: Indices used during formulation of the ILP.

Besides that, a lot of data is known about the tasks and analysts. The parameters used for them, are given in Table 4.2.

r_j	Release date task j
d_j	Due date task j
p_j	Processing time task j
M_{it}	Available time in hours of an analyst i on day t
y_{ij}	Analyst i has or hasn't the qualification to process task j
q_{jk}	Tasks j and k may or may not be proccesed on the same day by the same analyst

Table 4.2: Parameters used for the ILP-model.

For the parameters it holds that $r_j, d_j, p_j, M_{it} \geq 0$ and $r_j \leq d_j$. The parameters y_{ij} and q_{jk} are binary parameters.

If the variable y_{ij} is equal to one, analyst i is qualified to process task j, otherwise $y_{ij} = 0$. The parameter q_{jk} indicates if task j and task k may be processed on the same day by the same analyst ($q_{jk} = 1$) or not ($q_{jk} = 0$).

For assigning all the tasks to a specific day and analysts the variable x_{ijt} is introduced. This is a binary variable which indicates if a task j is executed by analyst i on day t ($x_{ijt} = 1$) or not ($x_{ijt} = 0$). This leads directly to the variable s_j, defined in (4.1), which represents the day on which task j is processed.

$$s_j := \sum_i \sum_t t x_{ijt} \quad \forall j \tag{4.1}$$

Now that all parameters and variables for the ILP-model are introduced, the restrictions are formulated. Each restriction is given and explained individually.

Every task has te be executed exactly once.

$$\sum_i \sum_t x_{ijt} = 1 \quad \forall j \tag{4.2}$$

Every analyst i is available for a certain time on each day t. Every task assigned to analyst i takes a part of the available time of the analyst. To assure that no more time is used then the time which is available for the analyst, the following constraint is used, where

$$P_{it} = \sum_j p_j x_{ijt}$$

defines the total workload for analyst i on day t.

$$P_{it} \leq M_{it} \quad \forall i, t \tag{4.3}$$

Moreover, the analyst also needs the qualifications to process the task.

$$x_{ijt} \leq y_{ij} \quad \forall i, j, t \tag{4.4}$$

When analyst i has not the qualification to process task j ($y_{ij} = 0$), for every day it must hold that $x_{ijt} = 0$. On the other side, if $y_{ij} = 1$, the value of x_{ijt} may be both 1 or 0.

Every task j has to be processed between the given release date and due date.

$$r_j \leq s_j \leq d_j \quad \forall j \tag{4.5}$$

The last constraint of the ILP-model indicates if two tasks may be processed by the same analyst on the same day.

$$x_{ijt} + x_{ikt} - q_{jk} \leq 1 \quad \forall (i, t), j, k \ (j < k) \tag{4.6}$$

4.3 Objective function

The objective function used for this research will be represented by V. It divides the workload in such a way over all the analysts that the number of analysts that is available for projects is maximized and the spare time over the remaining analysts is clustered as much as possible. The objective V is defined as follows:

$$V := \max \sum_i \sum_h \sum_t P_{iht}$$

where P_{iht} defines the absolute difference in workload between two analysts:

$$P_{iht} = |P_{it} - P_{ht}|.$$

We chose to work with objective V as defined above for convenience, but note that other objective functions would have been possible, too. Let us here briefly comment on this choice.

It is clear that minimizing the objective function results in a schedule where the work load is as uniformly distributed over all analysts as possible. Maximizing the objective value V yields a schedule that creates as "much as possible" unbalance between workload of analysts, and thereby seems to model well the goals that we have in mind here. It is easy to see that maximizing V creates a solution that minimizes the number of used analysts when there are only two analysts available on a single day. Also when there are only three analysts available on a single day, maximization of the value V will yield a solution that leaves one analyst free, if possible which is used to prove NP-hardness in the next section. In Section 7.1, we demonstrate that maximization of V yields exactly the desired effect on typical practical data from Grolsch. In the subsequent section of this chapter, we prove that maximization of objective V is NP-hard. Note, however, that this NP-hardness is not an artefact of the specific objective function V chosen, but inherent in the qualitative definition of the overall goal of this research: whenever possible, the work should be performed with as few analysts as possible,

which entails the NP-hard, classical bin packing problem as a special case. In addition, the additional constraints and restrictions such as release times, availabilities, and qualifications, make the problem no easier. We refer to Chapter 10 for an additional discussion together with the lexmax-function that seems to create similar schedules as V.

Because the function V is not linear, a standard linearization can be used. Therefore, a set of new constraints is added to the model, to ensure that each P_{iht} takes the correct absolute value. This are the constraints (4.7)–(4.10), where γ_{iht} indicates if $P_{it} < P_{ht}$ or not and M is a large enough constant.

$$\begin{aligned} P_{iht} &\geq P_{it} - P_{ht} && (4.7)\\ P_{iht} &\geq P_{ht} - P_{it} && (4.8)\\ P_{iht} &\leq P_{it} - P_{ht} + M\gamma_{iht} && (4.9)\\ P_{iht} &\leq P_{ht} - P_{it} + M(1 - \gamma_{iht}) && (4.10) \end{aligned}$$

Combining (4.7)–(4.10) gives the required equation

$$P_{iht} = |P_{it} - P_{ht}|.$$

Penalty cost for unscheduled tasks

When solving the integer linear programming problem with the algorithms described later in this report, a penalty cost is used for each task that is unscheduled. If this is the case, a penalty cost C is added to the objective value. The value of C is defined as follows:

$$C = -2m \max_j p_j.$$

When a task j is removed and placed on the list of unscheduled tasks, this decreases the value P_{it} for an analyst i and day t with the value p_j. In the worst case scenario, this value increases the objective function with p_j twice for each analyst, because V is symmetric. Using C as defined above, the advantage of unscheduling a task is compensated.

4.4 NP-hardness

We define GROLSCH as the problem defined in Section 4.2. When taking an instance of the problem with only one day, equal values M_i for each analyst i, $r_j = d_j = 1$ and $Q = \emptyset$ the problem reduces to the classical bin packing problem [7]. The variant with two analysts on one day and a total available workload of $2M$ is equal to the partition problem [7]. Therefore we can say that the problem GROLSCH is NP-hard even though the practical assignment entails no qualitative definition of an objective function. In the remaining of this section, we show that the problem GROLSCH in combination with maximizing the objective function V is also NP-hard by showing that maximizing V solves the partition problem if a solution to the partition problem exists.

We first simplify the problem to a single day with three analysts and a set of tasks $T = \{1, \ldots, N\}$, each with duration p_j for each task $j = 1, \ldots, N$. We assume that the maximum

workload M at each analyst is equal and satisfies $M = \frac{1}{2}\bar{P}$ where $\bar{P}$ denotes the total duration of all tasks:

$$\bar{P} = \sum_{j=1}^{N} p_j.$$

All other constraints are removed and we present a solution P as follows:

$$P = (P_1, \ldots, P_n),$$

where P_i is the work load at analyst $i = 1, \ldots, n$. In all that follows we assume for simplicity of notation that the orders are sorted so that

$$P_1 \geq P_2 \geq \cdots \geq P_n.$$

This allows us to rewrite the objective function V:

$$V = \sum_{i=1}^{n} \sum_{h=1}^{n} |P_i - P_h| = 2 \sum_{i=1}^{n-1} \sum_{h=i+1}^{n} (P_i - P_h).$$

Theorem The problem to maximizing V for GROLSCH is NP-hard.

Proof We first show that maximizing the objective function V produces a solution with all tasks scheduled at two analysts if it exists and therefore minimizes the number of analysts necessary to perform all tasks. Suppose a solution P exists with two analysts, then it holds that

$$V(P) = \frac{1}{2}\bar{P} + \frac{1}{2}\bar{P} = \bar{P},$$

because $P_1 = P_2 = \frac{1}{2}\bar{P}$ and $P_3 = 0$.

Now assume a solution P' exist that maximizes $V(\cdot)$ and $0 < P'_i \leq M$ for each $i = 1, 2, 3$. We will derive a contradiction to maximiality of V. We denote the work load for the last analyst with Δ, so that $\Delta = P'_3 > 0$. Then

$$P'_1 - P'_2 < \Delta,$$

otherwise P' is not optimal because assigning workload Δ to the second analyst results in V increasing with an amount of $2\Delta > 0$. We now have for the objective value $V(P')$:

$$\begin{aligned} V(P') &= (P'_1 - P'_2) + (P'_1 - P'_3) + (P'_2 - P'_3) \\ &< \Delta + (P'_1 - P'_3) + (P'_2 - P'_3) \\ &= \Delta + ((\bar{P} - \Delta) - 2\Delta) \\ &= \bar{P} - 2\Delta \\ &< \bar{P} \\ &= V(P). \end{aligned}$$

This means that $V(P') < V(P)$ and this contradicts the optimality of P'. That means that maximizing V cannot be done in polynomial time, as otherwise we would be able to solve the NP-hard decision problem PARTITION in polynomial time. This is impossible, unless P=NP.

Chapter 5

Scheduling algorithm

Because the computational time of the CPLEX-solver and the fact that the department of Quality Control is not able to purchase such a solver, other ways to construct solutions have to be found to solve the integer linear programming problem. It is necessary that these solutions could be implemented in each programming code and have a computational time that will be small enough to be used in practice. Generating solutions fast gives also the possibility to compare solutions if necessary in future research.

In the first section three possible local search algorithms are explained to implement on the problem. Restrictions on computational time and space are taken into account to find a suitable algorithm. Generating a starting solution will be explained in Section 5.2 and after that the selected algorithm is explained.

5.1 Local search algorithms

Three different local search algorithms are introduced in Aarts et al. [1]. Those are simulated annealing, tabu search and genetic algorithms. A short explanation will follow on each variant.

Simulated annealing was introduced by Kirkpatrick et al. [6] in 1983. This algorithm is based on the physical processing of annealing metal. The part of local search is derived from the cooling technique of the metal. The molecules in the metal are interacting with eachother to finally reach a steady state, called freezing. The cooling of the metal is translated into a walk between feasible solutions for the ILP with so called neighborhoods, which are feasible solutions obtained by making changes in the current feasible solution. Each of these neighborhoods is accepted as the new feasible solution with a certain probability, depending on the temperature at that moment. This temperature follows, similar to the cooling of metal, a cooling schedule with an initial and final temperature.

Simulated annealing is applied on a wide range of mathematical problems and turned out to be very succesfull. Under certain conditions the algorithm finds the optimal solution when an infinite number of iterations is performed. However, finding optimal solutions takes exponential time and therefore simulated annealing is mentioned as an approximation algorithm with high performance results.

The second local search algorithm is tabu search, introduced by Glover in 1989 [3] and 1990 [4]. It has similarities with simulated annealing but tabu search, as the name already indicates, deals with a tabu list. It can also be seen as a walk through feasible solutions were each possible neighbor is accepted with a certain probability. During the walk, the algorithm

places solutions on the tabu list with low values, to prevent cycling or leaving local optima. The solutions on the tabu list have an acceptance probability of zero. A disadvantage of tabu search is the computational space and time needed for the tabu list and checking the solutions every iteration.

Introduced by Holland [5] in 1975 genetic algorithms are based on evolution theory and population genetics. The mathematical variant starts with a population of initial solutions. Combining solutions and generating offspring with the best solutions found so far, creates new solutions each iteration and finally leads to an optimum.

Because of the computational advantages and the performance on practical problems, simulated annealing is implemented for the problem. Tabu search and genetic algorithms require a lot of computational space to store a lot of solutions which are in the tabu list in tabu search or defined as the population in the genetic algorithms.

5.2 Starting solution

Without a starting solution the algorithm might need a lot of time to find a feasible solution. Therefore the construction of a starting solution is defined in this section. Furthermore, the performance of the algorithm with a starting solution could be compared to the performance of starting from scratch.

Algorithm 1 gives the pseudocode for generating a starting solution. In the first two loops, all tasks are scheduled as early as possible. This means that the task is assigned to the first free analyst and day. The schedule created with Algorithm 1 has the property of using as much time of combinations of single days and analysts as possible, which goes towards the preferred solution.

Sorting list of tasks

To generate a starting solution, the list of tasks is sorted in multiple ways. The first sorts are sorting on processing time, release date and due date. The other is sorting on number of qualified analysts per tasks and is defined with α_j for each tasks j as follows:

$$\alpha_j = \sum_i y_{ij},$$

where y_{ij} indicates if analyst i is capable to perform task j. Sorting on the four parameters is done in both descending and ascending order.

5.3 Simulated annealing

Algorithm 2 gives the pseudocode for the implementation of simulated annealing. The most important factors of simulated annealing are the intital temperature, the cooling schedule and the neighborhoods with their acceptance probabilities. First of all, the neighborhoods will be defined. The acceptance probability and possible cooling schedules are presented in the last two sections.

Each starting solution consists of a list of scheduled and unscheduled tasks. All other lines of the algorithm will be explained during the next sections.

Algorithm 1 Starting solution

Input set of all data
sort list of tasks in predefined way
for all $j|r_j = d_j$ **do**
    find first free combination of analyst and day that satisfies all constraints
    schedule task for that analyst and day
    update parameters and variables
end
for all $j|r_j \neq d_j$ **do**
    find first free combination of analyst and day that satisfies all constraints
    schedule task for that analyst and day
    update parameters and variables
end
Output starting solution

Algorithm 2 Simulated annealing

Input set of all data and possible starting solution
set parameters for simulated annealing
while stopping criteria are not satisfied **do**
    choose one of the possible neighbour solutions
    calculate new objective value of the neighbour
    calculate acceptance probability
    accept or decline solution due to the acceptance probability
    if neighbour is accepted as new solution **then**
        update current solution
    end
    update parameters simulated annealing
end
Output final solution with objective value

5.3.1 Neighborhoods

Given a solution, a number of neighbors is defined. These neighbors are obtained by making changes in the current solution. Five neighbors are defined in this research and are explained in the list below and illustrated in Figures 5.1–5.5. The figures are showing four random combinations of a day and an analyst on the horizontal axes. The vertical axes denotes the total workload and the blocks represent tasks. The availability of an analyst is indicated with the bars at the top. The blocks in the right of the picture are unscheduled tasks.

Of course only solutions that are feasible are taken into account as neighbors of the current solution.

1. **Swap**: select two tasks and swap them in the schedule. This means that the analyst and day of the first task became the analyst and day for the second task and vice versa. It is possible that the analyst or the day remains the same.

2. **Move**: a random task from the schedule, is moved to another day or analyst, or both.

3. **Schedule**: plan a task from the list of unscheduled tasks if there is enough time available at the analyst and day.

4. **Insert**: select a task from the list of unscheduled tasks and find a qualified analyst and feasible day. Remove as few tasks as possible until the availability constraint is satisfied.

5. **Remove**: remove a scheduled task.

It is easy to see that the *insert* neighborhood is a combination of one or multiple *remove* neighbors with a *schedule* neighbor. This neighbor is defined to ensure that unscheduled tasks are scheduled, even if the space for that specific tasks is blocked by a high number of tasks which otherwise have to be moved one by one.

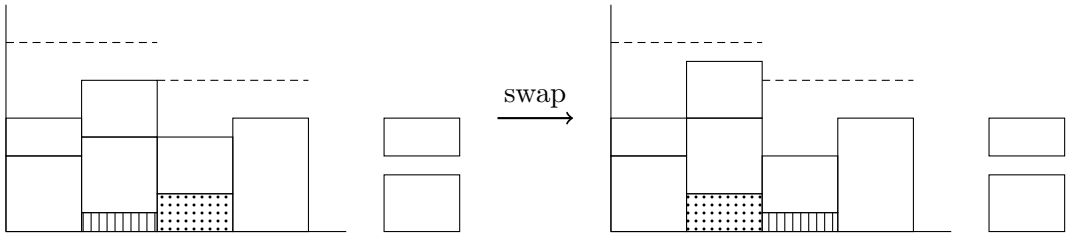


Figure 5.1: Illustration of *swap* neighborhood.

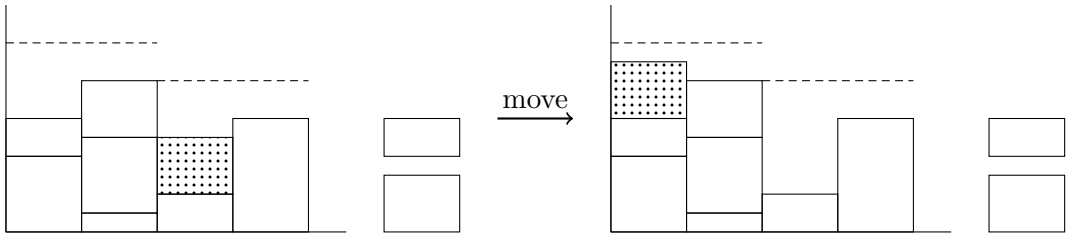


Figure 5.2: Illustration of *move* neighborhood.

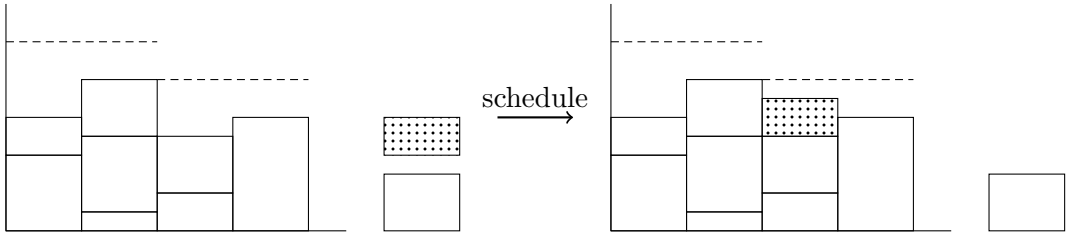


Figure 5.3: Illustration of *schedule* neighborhood.

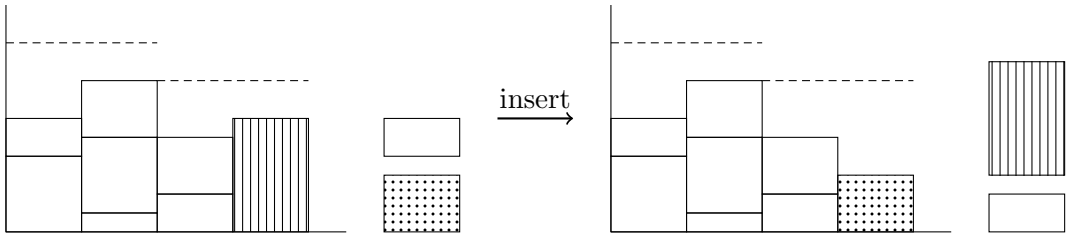


Figure 5.4: Illustration of *insert* neighborhood.

5.3.2 Acceptance probability

The probability of accepting a neighborhood solution depends on the difference in objective value between the current solution (S), the neighbor solution (N) and the temperature T_k for iteration k. Simulated annealing always accept a new neighbor as solution if it has a higher objective value. Otherwise, the neighbor is accepted with a certain probability. Mathematically the probability P_k for accepting the investigated neighbor at iteration k is defined as

$$P_k(\text{accept neighbor}) = \begin{cases} 1 & V(N) \geq V(S) \\ \exp\left(\frac{V(N)-V(S)}{T_k}\right) & V(N) < V(S) \end{cases}.$$

The choice of the initial temperature T_0 and the cooling schedule are playing a crucial role in the accepting of neighborhood solutions. A high temperature will lead to the acceptance of a high number of solutions, also solutions with a low objective value. As the temperature is cooling down, the probability of accepting a solution with a lower objective value becomes small. Because solutions with a higher value are always accepted, this lead to almost only solutions with a increasing objective value with respect to the feasible solution found so far. The algorithm terminates when the final temperature $T_f = 0.0001$ is reached. In this situation, the algorithm reaches the so called freezing state.

5.3.3 Cooling schedules

The cooling schedule of the simulated annealing algorithm defines the behaviour of the temperature from the annealing phase until the solution reaches the final temperature which is called the freezing state. The temperature T_k is changed each iteration k according to a predefined cooling schedule. Aarts et al. [1] gives possibilities for static and dynamic cooling schedules which are also given in Schneider et al. [8]. The last one also has collected a number of advanced cooling schedules.

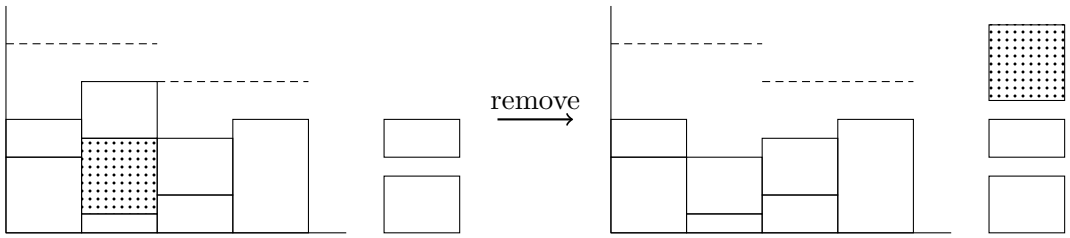


Figure 5.5: Illustration of *remove* neighborhood.

In this research the linear and exponential cooling schedule are used. The linear cooling schedule is defined as

$$T_{k+1} = T_k - \Delta T,$$

where $\Delta T > 0$ is the difference in temperature at each step, and

$$T_{k+1} = \beta T_k$$

defines the exponential cooling schedule with cooling factor $\beta \in (0, 1)$.

Initial temperature

Multiple ways of defining the inititial temperature T_0 are available. For most of them, a random walk is performed over all possible solutions by random chosing a solution defined as neighbor. The random walk gives a list of n objective values V_i for each solution $i = \{1, \ldots, n\}$. Two definitions of deriving an initial temperature are given below and the first one is used in this research.

The first definition of the initial temperature is presented in Aarts et al. [1] and defines the initial temperature as the maximal difference in objective value between solution i and $i + 1$:

$$T_0 = \max_{i=1,\ldots,n-1} |V_i - V_{i+1}|.$$

As mentioned in Aarts, the exact value of T_0 in this way, is hard to find, but a good estimation is found by performing a limited number of steps.

In Tarawneh et al. [9] a deviation average γ is used to define the initial temperature. This deviation is defined as

$$\gamma = \frac{1}{n-1} \sum_{i=1}^{n-1} |V_i - V_{i+1}|.$$

The size of the deviation is then linked to an interval from which the initial temperature is chosen at random.

Chapter 6

Data

The first section descripes the collecting of data over multiple systems at the brewery. In the next sections each time a group of datasets is explained. Details are given about the composure of the datasets and the purpose of the data is mentioned.

6.1 Collecting data

At the start of this research, almost no data of tasks at the laboratoy was available. Therefore, a lot of data from other systems at the brewery was collected, rearranged and combined to a number of datasets. Those datasets represent different periods with different numbers of tasks and analysts.

The most frequently used system for collecting all the data is LIMS (Laboratory Information Management System). This system contains almost all data of the tests which are processed by the Quality Control department. For example, numbers of tests are collected in this system. Except that, three analysts gave an indication of the processing times of all the tests processed by the analysts at the laboratory. Combining these processing times with the number of tests per task has given an estimate for the total processing time of a task.

For producing and packing the products, the brewery uses another system. This system gives the data about the number of switches of products produced and the number of lines that are in use for packaging the products. From this information a number of tasks and their processing times are reduced.

The list of tasks is completed with periodic tasks from the laboratory itself.

In almost all cases, an analyst is available for eight hours a day. The information about the availability of the analysts is obtained from the work schedules prepared by the manager. It will be taken into account later in this report that an analyst might have meetings and some other regular personal tasks each day, for example checking their mail and having seminars.

6.2 Historical datasets

Based on various sources of data in the brewery, five datasets representing the workload at the department of Quality Control were constructed. These datasets were constructed to generate the very first performance results with the integer linear programming model.

The group of data represents five weeks (41 to 45) of 2014, each containing a list of 302 tasks, 15 analysts and 5 days.

6.3 Datasets test week

Two test weeks were organized during the research period. Especially the first one is mentioned in this report. Even the best week found for the second test week, was not a very appropriate one. The main reason for this was the high number of interns and the low number of permanent analysts available. This lead to analysts with a fixed list of tasks each day independent of the qualifications and availabilities of other analysts.

The data for a test week was obtained from the different information and planning systems available. A first estimation of the list of tasks was made a week in advance and the last changes were made the weekend before, both in consultation with the manager and an experienced analyst. Also, during the test week, some of the comments directly lead to changes for the remaining tasks during the test week.

6.4 Datasets to analyze performance of simulated annealing

Table 6.1 gives an overview of the datasets used for the performance analysis of the simulated annealing algorithm presented in Chapter 5.

Dataset	Description	Tasks	Analysts	Days
1	Second test week	81	11	2
2	Example report	11	3	2
3	First test week	309	11	5
4	Combined data	302	15	5
6	Combined data	302	15	5
7	Combined data	302	15	5
8	Second test week	116	13	3

Table 6.1: Overview of datasets used for performance analysis of the algorithms.

Dataset 2 is the illustrative example given in Section 2.4 of this report. With only eleven tasks, three analysts and two days, it is the smallest dataset used. Also this dataset does not occur in a practical situation.

Datasets 1, 3 and 8 are derived from or are the same as the datasets used during one of the two test weeks. For example, dataset 3 is exactly the one used for scheduling the whole first test week. As can be seen in Table 6.1, datasets 1 and 8 are a part of the test week data.

Datasets 4, 6 and 7 are based on the datasets presented Section 6.2. This is data from a few weeks constructed from information given by other systems used in the brewery.

Daily tasks and releases as mentioned in Section 2.1 have a release and due date which are exactly the same. As the name already indicates, this means that there is only one day on which that specific task may be processed. A periodic task, as mentioned in the same section, has a due date which is later than the release date. So there are multiple days on which the task can be scheduled. In the datasets more then half of the tasks are daily tasks or releases. The rest of the tasks is periodical.

The processing time of each task over the datasets varies a lot. A few tasks are taking

only a few minutes to be processed. For example, the releases are scheduled on most days for ten to twenty minutes to distribute them over all the work places.

Otherwise, tasks have a processing time which is equal to the availability of the analysts for a day. Filtering monsters for microbiology is an example of such a task. It has to be done very often, so normally an analyst is busy with this task almost the complete day.

When an analyst is working the whole day, this means he is available for 6,5 hours despite the fact he is at the department for eight hours. The other time is for example spent on checking their mail and other additional work.

Chapter 7

Computational results CPLEX

This chapter presents the computational results for the different groups of datasets as well as the computational and practical results of the first test week. The interface AIMMS is used with the solver CPLEX 12.5.1 when solving the integer linear programming problem.

The results are presented in the same order as in the chapter where all the groups of data are explained.

7.1 Historical datasets

The results of the ILP for the five historical datasets are presented in Table 7.1. It can be seen that a large gap remains after solving the integer linear programming.

Dataset	Wk41	Wk42	Wk43	Wk44	Wk45
Iterations	9.255.515	12.122.497	6.057.478	10.380.991	9.333.494
Nodes	134.541	226.321	108.502	158.189	105.825
Nodes left	50.295	218.923	96.508	112.501	64.562
Best LP bound	263.910	240.701	212.972	224.912	255.525
Best solution	207.525	191.661	167.520	181.379	204.090
Gap (%)	27	26	27	24	25

Table 7.1: Results after running the solver for maximizing the objective function V (constraints: 3.461.511; variables: 26.805; integer variables: 24.077).

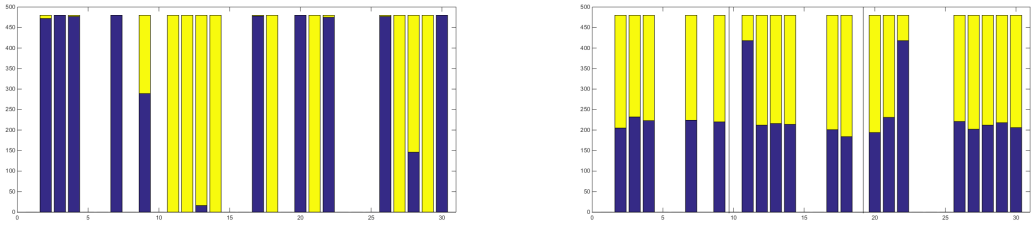
The value of 'Gap' in percentages gives to so called 'integrality gap' between the current feasible solution and upper bound found on the ILP so far. When the value of the gap is zero, this means that the solution found is proven optimal. Small values of the gap therefore means that the solution is almost optimal.

In this situation, the gap has values around 25 percent, what means that either the solution is far away from the optimal solution or the upper bound is very high and could be improved. However, when regarding the values corresponding to the feasible solutions found, it seems that the solution look almost optimal. In Figure 7.1 the visual representation is given of a selection of the days of a single dataset.

Figure 7.1a shows the results obtained when maximizing the objective function. As expected, when an analyst is performing a task, this means that as much tasks as possible are

scheduled by that analyst. Depending on the total processing time over all analysts scheduled on one day, there will be an analyst scheduled busy only a part of his total available time.

When the objective function V is minimized, the work load is almost equally distributed over all the available analysts, as been shown in Figure 7.1b. The two outliers are explained by the fact that those analysts are scheduled on tasks that take significantly more time then the average scheduled time for an analyst.



(a) Maximizing V

(b) Minimizing V

Figure 7.1: First computational results from CPLEX-solver in AIMMS. Graphics showing two out of five days from one of the described datasets. Blue indicates the scheduled time for an analyst, yellow time is not scheduled with tasks for an analyst.

7.2 Practical results test week

This section gives a short overview on the results of the test week and the accuracy of the data used. One of the reasons to organize a test week was to obtain information about all practical side effects of the new model, such as the arrangement of new schedules for the analysts.

The schedule for the test week was created with AIMMS and discussed with the manager. After this discussion, some conflicts where removed which occured due to imperfections in the data, i.e. the data was changed and a new schedule was created. Also during the test week a number of differences in processing times occured and were taken into account for the remaining days of the test week.

The total time available for an analyst on a day was estimated at 6,5 hours. After evaluating the test week, this estimation seems to be a good estimation. Almost all days, the analysts had enough time to perform all the tasks they were assigned to.

The Figures 7.2–7.4 give an indication of the first results of the testweek and the accuracy of the data. Each day of the testweek the analysts made notes about the processing times and those are compared with the processing times used before.

Figure 7.2 shows the difference between the scheduled time in advance and the time calculated after the testweek. The most remarkable is the high bar on monday, which is explained by the fact that this was the first day with the new schedule. Therefore, the analysts had to get used to the new schedule. Also the evaluation planned at the end of the day took some time.

Figure 7.3 indicates that on average eighty percent of the time is used for regular tasks during the test week. The remaining time could be scheduled for project hours.

Figure 7.4 represents the percentage of time that is not used for other appointments, such as weekly meetings and seminars.

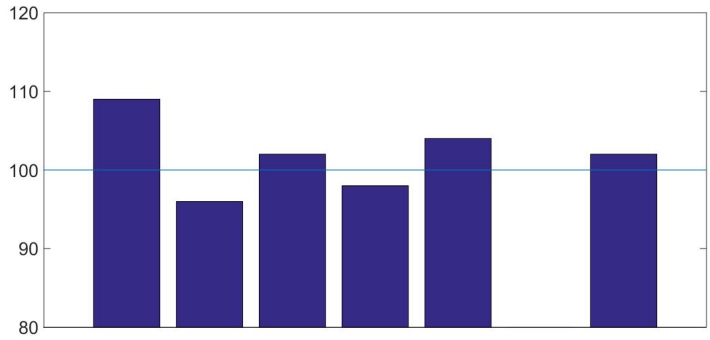


Figure 7.2: Percentage of real working time with respect to the scheduled time. The bars represent the days of the week and the weekly average.

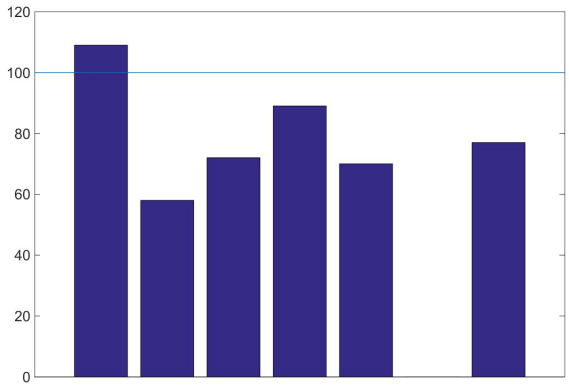


Figure 7.3: Percentage of time spent on performed tasks with respect to the total availability of the analysts. The bars represent the days of the week and the weekly average.

7.3 Simulated annealing: optimal values datasets

Using the CPLEX-solver in the AIMMS-environment, there was tried to find an optimal solution for the objective function for each dataset. Because of the large size of some datasets and the combination of constraints, only an upper bound was found for some of them. In Table 7.2 the values are given which are used for the performance analysis of the algorithms. Later on, there will be refered to as 'optimal values', however some of them are not proven to be optimal or even might not be optimal values either.

To make it easier for the CPLEX-solver to find a solution which is optimal or which is not far away from the optimal solution, a constraint on the objective value itself is introduced for some datasets. When this upper bound is added, a lot of possible solutions are disregarded by the solver, because the objective value will be too high. This decreases the possible solutions

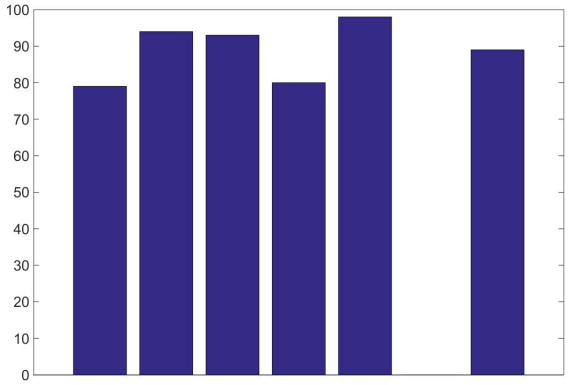


Figure 7.4: Available time to schedule for an analyst compared to the total time on the weekly schedule of the department. The bars represent the days of the week and the weekly average.

that have to be taken into account while searching for a feasible solution. In most cases, a feasible solution is found earlier in comparison to the situation where the upper bound is not used.

In the column 'Notes' information is given about the value of V. This value might be proven optimal. Otherwise, it is also mentioned if an upper bound is used.

Dataset	V	Notes
1	44.940	Value LP bound, no solution found
2	64	Proven optimal
3	109.674	V almost equal to LP bound
4	243.000	Upper bound used for V
6	238.904	Upper bound used for V
7	243.120	Upper bound used for V
8	92.720	Proven optimal

Table 7.2: So called optimal values for objective function V used for performance analysis.

Chapter 8

Computational results simulated annealing

This chapter gives an overview of the computational results of simulated annealing. The performance of running with different starting solutions is given in the first section. After that, results on the different settings of simulated annealing are presented. The initial temperature is determined with a random walk and the performance of two different cooling schedules is given. Visual representations of the objective values during a run with different cooling schedules are also given in Section 8.3.

All the performances are given in percentages of the 'optimal' values given in Section 7.3 and the computational time is given in seconds. Most of the times, dataset 3 is used for the presentation of the results. Equal results are obtained for almost all datasets. Each simulation is runned twenty times with equal settings in MATLAB. The optimal settings for simulated annealing are presented in the conclusion in Chapter 9.

Boxplots are used to give a visual representation of the results. The boxplots indicates the minimum and maximum value (except outliers), the 25th and 75th percentile are between the edges of the box and measurements considered as outliers are the (red) crosses.

8.1 Starting solution

To create a starting solution, eight different ways to sort the task list were mentioned in Section 5.2. The performance results of simulated annealing for each of those different starting solutions is given in Table 8.1.

Larger values for β are necessary to create feasible solutions in most cases. However, when the task list is sort on $-\alpha_j$ and p_j even the results with $\beta = 0.999$ are worse. There is even a large chance that solutions found are not feasible at all.

The best results are obtained with sorts α_j, $-p_j$ and $-r_j$. Only the first sort is already obtaining better solutions for lower values of the exponentital cooling factor. Therefore, this sort is used to perform the other runs.

Not only a better performance is obtained when using a starting solution. The computational time decreases significantly to reach those solutions. A number of fifty runs is performed to see the difference between simulated annealing starting from scratch and simulated annealing that begins with a starting solution with a task list sorted on the number of qualified analysts.

For the results of dataset 3 presented in Table 8.1 only one unfeasible solution is found

		β				
		0.8	0.9	0.95	0.99	0.999
α_j	Max	91.3	91.3	91.6	99.6	100.0
	Min	85.3	85.3	85.5	86.1	99.5
	Avg	86.1	86.4	88.6	92.0	99.7
$-\alpha_j$	Max	-2.2	25.3	42.4	70.8	99.6
	Min	-24.5	-18.9	-13.2	20.2	54.7
	Avg	-15.3	2.5	10.4	46.3	77.3
p_j	Max	55.6	57.7	57.6	78.6	94.3
	Min	38.4	44.4	44.8	59.7	67.2
	Avg	45.2	49.6	53.0	66.4	85.0
$-p_j$	Max	55.2	77.0	93.8	99.6	100.0
	Min	10.9	33.1	27.7	88.1	99.4
	Avg	35.5	49.6	57.4	95.8	99.8
r_j	Max	74.0	85.2	91.4	99.2	100.0
	Min	51.6	57.4	74.2	86.1	88.1
	Avg	61.7	73.2	84.5	91.9	98.6
$-r_j$	Max	68.9	80.5	91.4	98.9	100.0
	Min	51.6	57.5	63.2	86.1	99.5
	Avg	60.1	71.4	81.2	90.4	99.7
d_j	Max	73.9	91.0	90.9	99.6	100.0
	Min	51.9	58.2	58.2	85.9	88.1
	Avg	64.3	73.9	80.8	94.0	99.2
$-d_j$	Max	74.6	85.6	91.2	99.6	100.0
	Min	52.2	51.6	63.2	82.4	87.9
	Avg	66.6	76.1	82.4	91.3	98.0

Table 8.1: Performance for eight starting solutions with five values for the exponentital cooling factor.

when using the starting solution in contrast to 28 unfeasible solutions when starting from scratch. The average time for a run from scratch is almost five times larger than a run with starting solution, a run from scratch took about one minute on average. Remarkable is the fact that if feasible solutions are found the performance is similar between starting from scratch and starting with a starting solution.

8.2 Initial temperature

The initial temperature is set by performing a random walk over the neighborhoods. The maximum difference in objective value between two of those neighbors will be the value of the initial temperature T_0. In order to determine the influence from the number of steps in the random walk on the performance of simulated annealing, different runs are performed. Table 8.2 and Figure 8.1 show the results for runs with three different values of the exponential cooling factor and six different sizes of the random walk.

The results show a lot of deviation in the objective values for $\beta = 0.95$ and $\beta = 0.99$. The

		β					
		0.95		0.99		0.999	
Steps		%	t	%	t	%	t
10	Max	91.5	0.2	99.4	1.0	100.0	12.4
	Min	85.7	0.1	86.1	0.5	99.5	9.9
	Avg	87.7	0.1	94.0	0.7	99.8	10.9
50	Max	92.0	0.3	99.5	0.9	100.0	12.5
	Min	81.5	0.1	86.0	0.6	99.5	9.7
	Avg	87.2	0.1	92.4	0.6	99.8	11.4
100	Max	91.4	0.2	99.5	1.2	100.0	13.7
	Min	85.4	0.1	86.2	0.6	88.3	7.1
	Avg	88.2	0.1	93.6	0.7	98.6	11.6
250	Max	91.4	0.2	99.6	1.1	100.0	14.3
	Min	85.4	0.1	85.9	0.5	93.5	6.29
	Avg	87.7	0.1	92.7	0.7	98.6	11.4
500	Max	91.5	0.2	99.6	1.1	100.0	14.9
	Min	85.3	0.1	85.7	0.6	88.0	6.1
	Avg	88.7	0.1	92.1	0.7	98.0	11.2
1000	Max	91.5	0.2	99.8	1.2	100.0	14.3
	Min	85.3	0.1	85.7	0.6	93.8	7.4
	Avg	87.6	0.1	93.9	0.8	99.0	12.3

Table 8.2: Performance when using different numbers of steps in the random walk and different cooling factors for the exponential cooling schedule.

results become stable for runs with $\beta = 0.999$. However, a short random walk is necessary, see for example the results in Figure 8.1 with $\beta = 0.999$ and ten and fifty steps in the random walk. When increasing the number of steps, outliers appear. Those could be explained by the fact that a longer random walk results in a higher initial temperature. This means that solutions with lower objective values have a higher chance of becoming a new current solution at the beginning of the run. It seems that with a cooling factor of $\beta = 0.999$ the run is terminating too early in some cases, creating the outliers. Of course this problem could be solved by using a higher value for β, but that in turn increases the computational time.

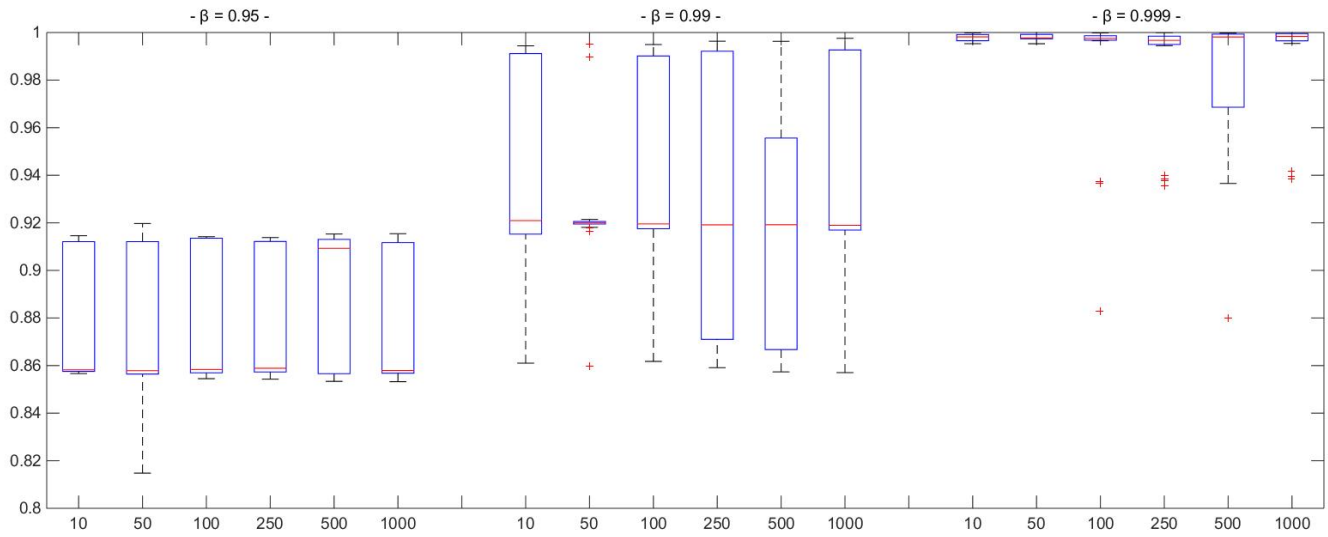


Figure 8.1: Performance when using different numbers of steps in the random walk and different cooling factors for the exponential cooling schedule.

8.3 Cooling schedules

The performances of two different cooling schedules are presented in this section. The results for the linear cooling schedule are presented in Table 8.3 and Figures 8.4–8.5. The results for the exponential cooling schedule are presented in Table 8.4 and Figures 8.6–8.7. The results for the second dataset are not presented in graphical form, because this situation does not occur in practice.

Feasible solutions are found in almost all cases for the exponential cooling schedule with $\beta = \{0.99, 0.999\}$ and the linear cooling schedule with $\Delta T = \{0.05, 0.01\}$. The computational time to construct those solutions, is more for the linear cooling schedule in comparison with the exponential cooling schedule. This is explained by the fact that the number of iterations for the linear cooling schedule is significantly higher. To compare, an exponential cooling schedule with $\beta = 0.999$ is equivalent to a linear cooling schedule with $\Delta T \approx 0.06$ when looking at the investigated neighbors. The linear cooling schedule therefore accepts bad solutions with a higher probability then the exponential one. Figures 8.2 and 8.3 show respectively a run of the exponential and the linear cooling schedule. Note that the number of iterations of the linear run is five times higher than the exponential one where approximately the same objective values are obtained.

The run with a linear cooling schedule shows a lot of fluctuations. This is caused by a reasonable chance to accept a solution with a lower objective value, also on the long run. This is in opposite with the exponential cooling schedule where only at the beginning a number of fluctuations is shown.

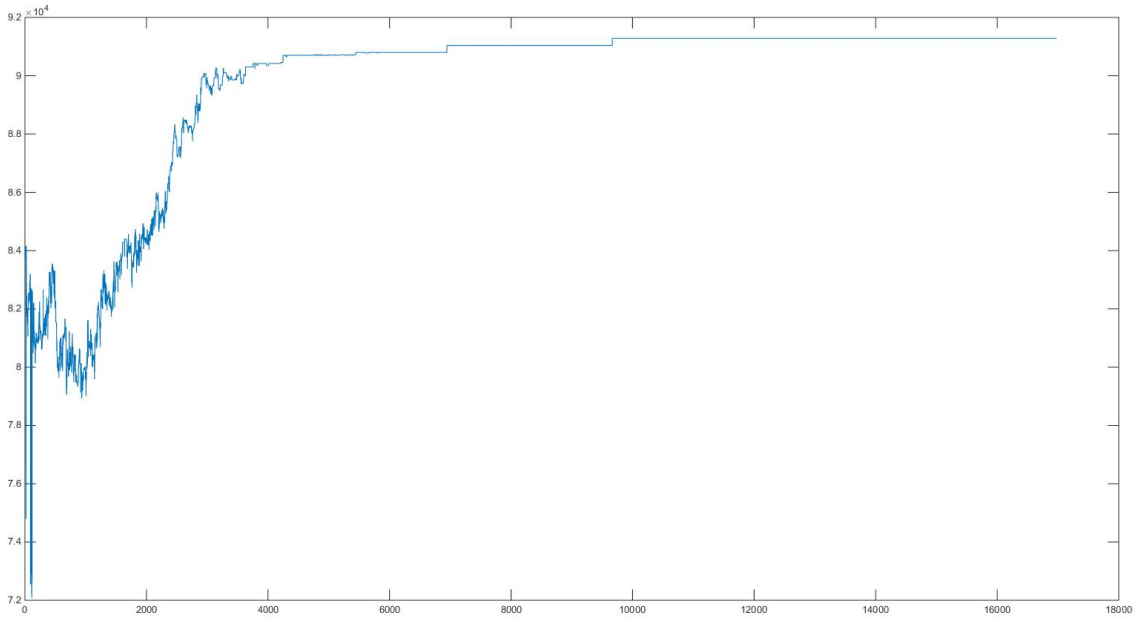


Figure 8.2: Value of objective function for a run with an exponential cooling schedule.

It has to be mentioned that for the first dataset, CPLEX and simulated annealing were not

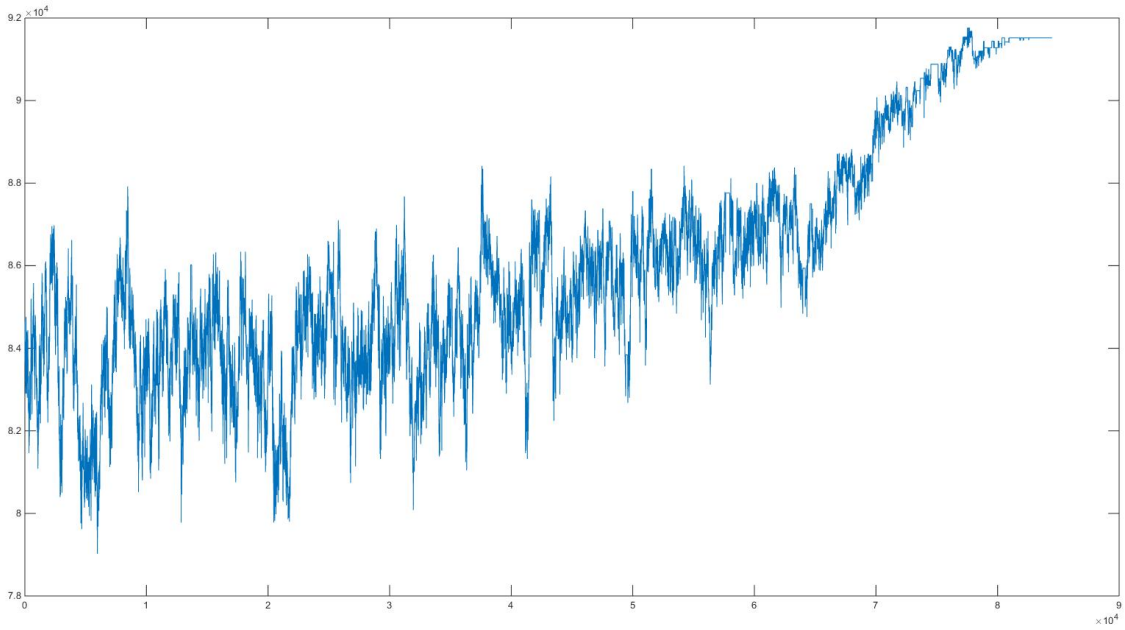


Figure 8.3: Value of objective function for a run with a linear cooling schedule.

able to found a feasible solution. A number of two tasks remain unscheduled when using the settings above for all runs. Because this dataset is constructed out of data of the second test week, this sounds strange. However, only two days were used and some tasks therefore had to be scheduled over a smaller interval. For example, tasks that may be executed over the whole test week, must be executed in two days in the constructed dataset.

Concluding remarks

In summary, our computational remarks show the following:

- using a starting solution decreases the computational time significantly;
- a satisfying initial temperature is already obtained with a short random walk if less computational time is available;
- the fastest way to obtain solutions with simulated annealing in this cases is with an exponential cooling schedule.

		ΔT											
		1		0.5		0.25		0.1		0.05		0.01	
Dataset		%	t	%	t	%	t	%	t	%	t	%	t
1	Max	84.8	0.6	84.7	1.3	85.2	2.5	85.6	4.7	85.3	9.0	85.3	50.9
	Min	80.0	0.0	82.4	0.0	79.8	0.2	82.2	0.1	82.3	2.6	81.9	24.2
	Avg	83.4	0.2	83.7	0.4	83.7	1.3	83.9	2.8	84.1	6.4	84.3	37.5
2	Max	100.0	0.1	100.0	1.2	100.0	1.2	100.0	6.8	100.0	10.0	100.0	43.1
	Min	62.5	0.0	62.5	0.0	62.5	0.0	75.0	0.0	87.5	0.0	100.0	1.0
	Avg	83.8	0.0	83.1	0.2	85.0	0.3	95.0	1.1	98.1	1.9	100.0	8.9
3	Max	91.2	0.4	99.1	2.0	99.4	4.5	99.7	7.1	99.8	24.0	100.0	253.8
	Min	85.1	0.0	85.0	0.0	85.1	0.1	85.6	0.1	85.4	0.0	99.4	1.9
	Avg	87.3	0.2	88.3	0.4	91.7	0.8	96.5	2.0	98.6	4.8	99.8	30.7
4	Max	97.4	1.3	97.8	2.8	97.3	6.9	98.7	19.8	98.9	25.0	99.8	955.6
	Min	91.7	0.3	94.7	0.4	94.8	1.0	96.6	2.0	97.8	4.9	99.1	27.8
	Avg	95.5	0.6	96.0	1.0	96.2	2.8	97.4	5.0	98.4	8.0	99.5	159.1
6	Max	90.9	1.6	91.2	2.8	92.3	5.1	93.8	15.4	94.2	28.2	95.1	620.8
	Min	86.7	0.2	88.7	0.4	89.3	1.5	90.3	2.1	91.8	4.5	94.5	34.1
	Avg	88.9	0.7	89.9	0.9	90.6	2.3	92.5	6.2	93.5	10.9	94.8	159.8
7	Max	96.9	1.4	98.0	2.6	97.5	5.7	98.6	16.1	99.1	35.6	99.9	356.5
	Min	93.4	0.3	95.5	0.5	96.1	1.2	97.6	2.5	98.2	6.2	99.3	27.2
	Avg	95.8	0.6	96.4	1.0	97.0	2.3	97.9	6.0	98.6	16.4	99.6	111.2
8	Max	94.9	1.5	95.5	3.2	96.8	4.4	98.4	11.8	99.0	29.7	99.4	287.5
	Min	90.0	0.2	90.6	0.6	93.8	0.9	95.1	3.7	94.3	5.9	97.5	42.2
	Avg	93.3	0.7	93.7	1.4	95.4	2.4	96.9	6.5	97.4	12.9	98.6	80.4

Table 8.3: Performance of linear cooling schedule for different values of ΔT for each dataset.

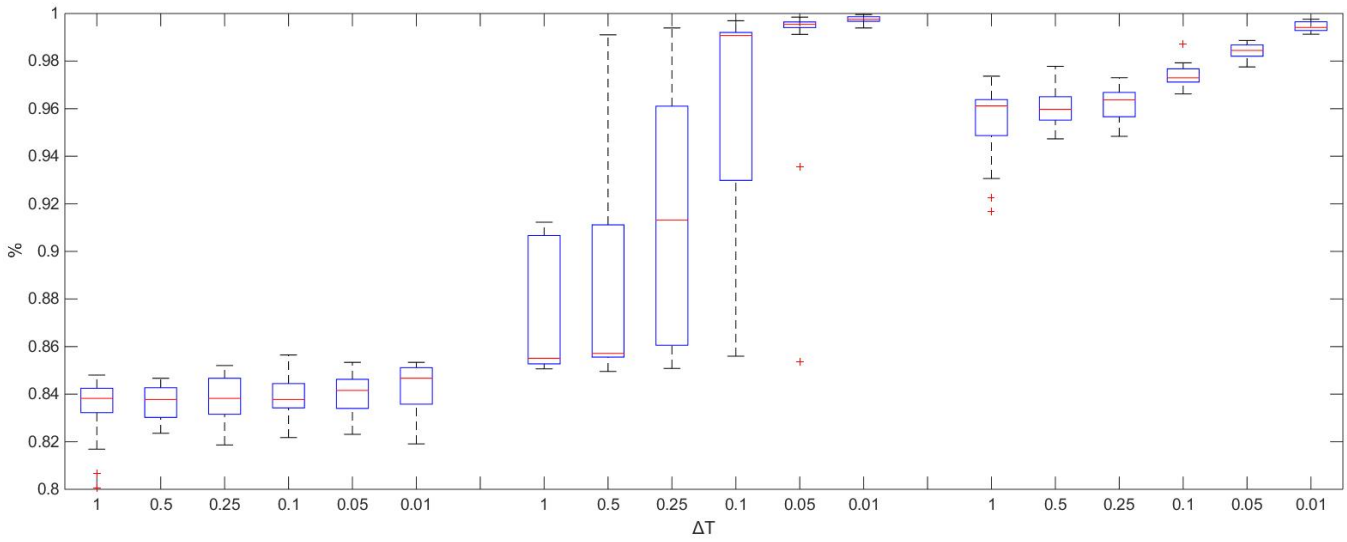


Figure 8.4: Performance of linear cooling schedule for different values of ΔT for datasets 1, 3 and 4.

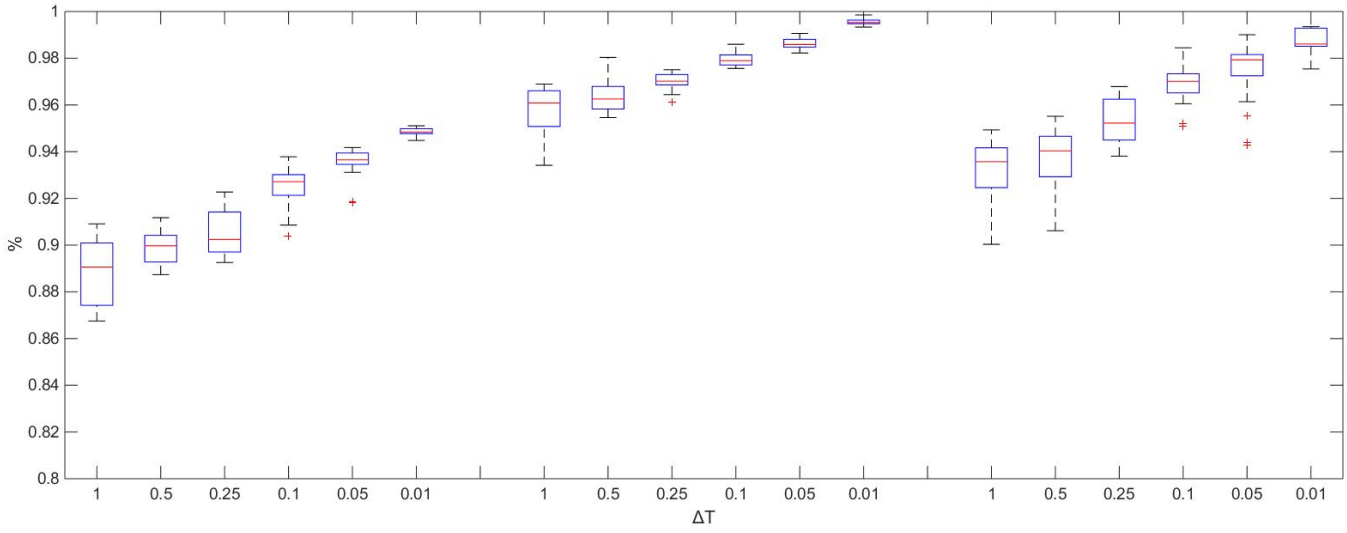


Figure 8.5: Performance of linear cooling schedule for different values of ΔT for datasets 6, 7 and 8.

		β									
		0.8		0.9		0.95		0.99		0.999	
Dataset		%	t	%	t	%	t	%	t	%	t
1	Max	84.9	0.0	85.0	0.4	85.1	0.9	85.1	3.5	85.7	36.0
	Min	82.8	0.0	83.2	0.0	82.2	0.1	81.4	0.3	82.4	2.9
	Avg	83.9	0.0	84.2	0.1	84.1	0.2	83.6	2.0	83.5	27.6
2	Max	100.0	0.4	100.0	0.5	100.0	2.1	100.0	85.3	100.0	77.5
	Min	75.0	0.0	75.0	0.0	87.5	0.0	87.5	0.2	87.5	12.0
	Avg	93.8	0.1	95.6	1.0	99.4	0.3	99.4	9.9	99.4	26.8
3	Max	91.3	0.0	91.3	0.1	91.3	0.2	99.5	1.1	100.0	14.5
	Min	85.2	0.0	85.3	0.1	85.3	0.1	86.3	0.6	93.5	6.9
	Avg	86.3	0.0	87.0	0.1	87.7	0.1	91.7	0.7	99.4	12.3
4	Max	98.7	0.1	98.5	0.1	99.0	0.2	99.4	0.9	99.7	9.1
	Min	98.0	0.0	98.2	0.1	98.1	0.2	98.6	0.7	98.8	7.7
	Avg	98.3	0.0	98.3	0.1	98.5	0.2	99.1	0.9	99.4	8.4
6	Max	91.7	0.0	91.9	0.1	92.0	0.2	92.8	0.6	95.1	5.9
	Min	90.6	0.0	91.0	0.0	91.2	0.1	90.8	0.5	91.2	4.7
	Avg	91.3	0.0	91.4	0.1	91.6	0.1	92.0	0.5	93.0	5.2
7	Max	98.0	0.1	98.3	0.1	98.5	0.3	99.2	1.0	99.8	9.7
	Min	97.6	0.0	97.6	0.1	97.8	0.2	98.3	0.8	98.9	8.3
	Avg	97.9	0.1	98.0	0.1	98.2	0.2	98.9	0.9	99.4	8.9
8	Max	92.3	0.4	93.1	1.0	93.7	1.6	96.1	7.6	98.8	70.0
	Min	90.1	0.0	90.9	0.1	90.8	0.1	90.8	0.4	90.9	4.3
	Avg	91.2	0.1	91.8	0.2	92.4	0.4	93.1	2.2	96.0	23.0

Table 8.4: Performance of exponential cooling schedule for different values of β for each dataset.

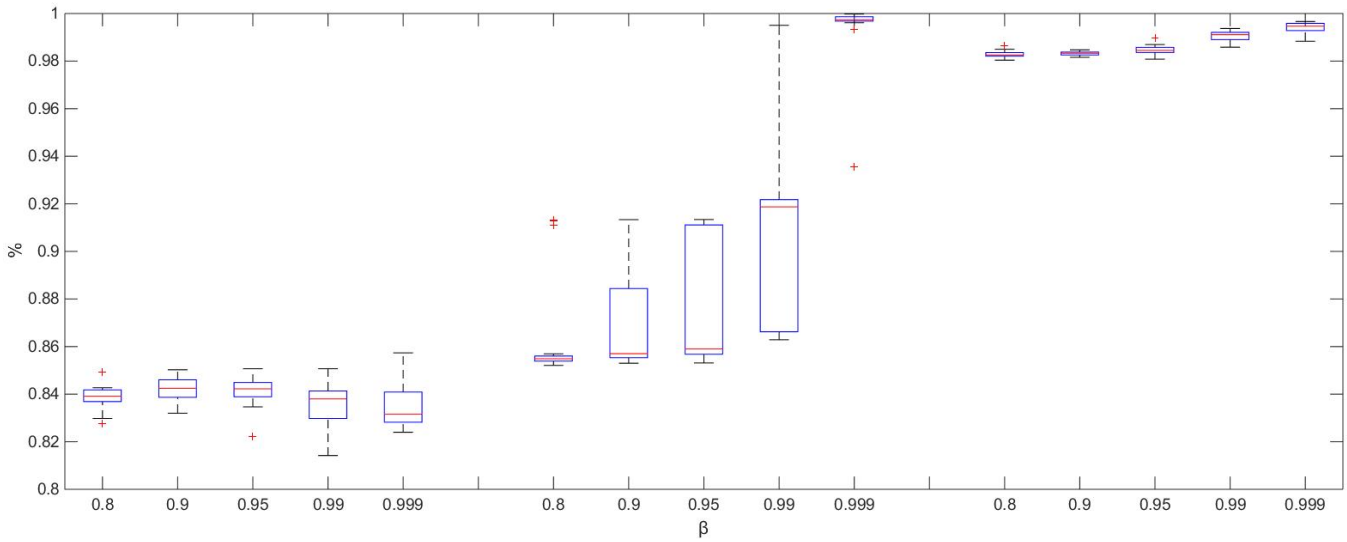


Figure 8.6: Performance of exponential cooling schedule for different values of β for datasets 1, 3 and 4.

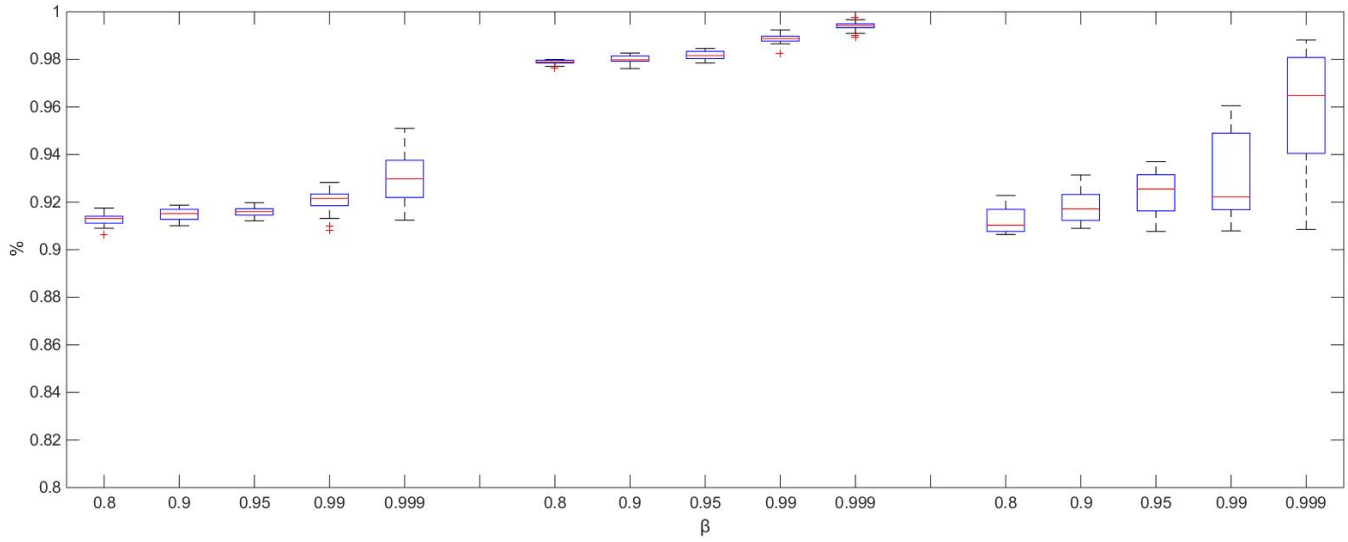


Figure 8.7: Performance of exponential cooling schedule for different values of β for datasets 6, 7 and 8.

Chapter 9

Conclusions

This research was performed to investigate the possibilities for a new scheduling method at the department of Quality Control. The method needs to minimize the number of analysts necessary for all tasks and also produce a certain profile over the busy analysts as presented in the problem description. Also other restrictions have to be satisfied and it has to be possible to implement the algorithm in a simple tool without large costs and the method should not be time consuming.

Due to the cost and time criteria, a solver such as CPLEX does not fulfills the requirements. Therefore the local search algorithm simulated annealing was used and the performance was analyzed for the datasets representing the practical situation. It turned out that with the right settings, simulated annealing produces schedules with solutions between 90 and 100 percent of the optimal values. Less computational time is necessary to find solutions with simulated annealing and the algorithm can be implemented in almost any programming language.

The best solutions, taking in consideration computational time, are obtained with an exponential cooling schedule with cooling factor $\beta = 0.999$. A short random walk of fifty steps is already giving a good initial temperature if a short computational time is available. Furthermore, a starting solution decreases the computational time of a run with simulated annealing significantly. The best option for sorting the task list, is sorting the tasks from a low number of qualified analysts to a high number of qualified analysts to perform each task.

Of course, when more time is available, it is recommended to use other settings where longer runs are performed resulting in better solutions. The cooling factor β can be increased and also a longer random walk could be performed, both increasing the computational time. Increasing the cooling factor decreases the possibility of unfeasible solutions at the end of a run, because the chance to accept a worse solution becomes very small.

A test week was hold twice to see the practical influence and issues of the new scheduling method. Overall the new method was received well, but of course the analysts need to get used to the new scheduling method were they left the current idea of work places. Talking about savings in hours for projects is hard and that is a point that has to be researched on the long run when the new scheduling method will be implemented.

More information about implementing the scheduling method in practice together with practical issues that might occur, are given in the recommendations.

Chapter 10

Discussion and recommendations

Availability of data and finetuning parameters

A total of seven datasets was used during this research. To create this datasets from scratch was not an ideal situation. There was tried to construct datasets that gave a good representation for the practical situation at the department of Quality Control. However, using CPLEX and simulated annealing did not result in a feasible solution for the first dataset. Having a large number of datasets representing the workload at the department, will lead to results which are better supported. This also means that the parameters of the simulated annealing algorithm could be estimated better.

Further research

Instead of using simulated annealing, also the performance of the two other algorithms mentioned could be analyzed. Those algorithms, tabu search and genetic algorithms, are also used for a number of practical issues, but as already mentioned, they take more computational space than simulated annealing.

When using simulated annealing in practice, all the parameters could be estimated better when having more datasets available. This also means that the results of the simulated annealing algorithm, could be checked with the practical implementation. At this moment, we do not know how many times simulated annealing will be used to create a schedule. It could be necessary on a daily basis or on a weekly basis and also the length of the planning horizon might be different each time. All those factors may have effect on the settings of simulated annealing.

Two other topics remained open during this research. The first one is the effect on the objective value when some daily tasks have to be scheduled at the same analyst on a weekly basis. Maybe this will lead to a huge impact on the objective value, but in practice it could be easier for the analysts to work on the same daily tasks each week.

The second one is variation in tasks for the analysts. With the current method, there is a change that an analyst performs certain tasks each day or week. When using some kind of variation function on the solution found by simulated annealing, this may lead to better variation in tasks over the analyst. Another option is to define neighborhoods depending on the variation in tasks, rather then on only the total work load at an analyst.

Objective function

So far, it is not proven that the function V minimizes the number of analysts and therefore works correctly. However, with all the simulations and intuition we had so far it gave the desired solution and a counter example is not found untill now. It seems to be that V is minimizing the number of analysts necessary to schedule all tasks and finds a solution where the remaining time is as unfragmented as possible.

Another objective function, that seems to do the same as V is the lexmax-function. This function first maximizes the load at the first analyst, then at the second, and so on. It sorts vectors in their lexicographic order, which could be compared to sorting words alphabetically. But with a counterexample, it is proven that this function does not minimize the number of analysts in all cases.

With an avaibility of 1 for each analyst, n tasks of size $1 - \frac{3}{2n}$ and n tasks of size $\frac{1}{n}$, the lexmax-function creates a solution with $n+1$ analysts. The solution with lexmax is presented in Figure 10.1 on the left. However, the optimal solution with respect to minimizing the number of analysts uses n analysts, as presented in Figure 10.1 on the right.

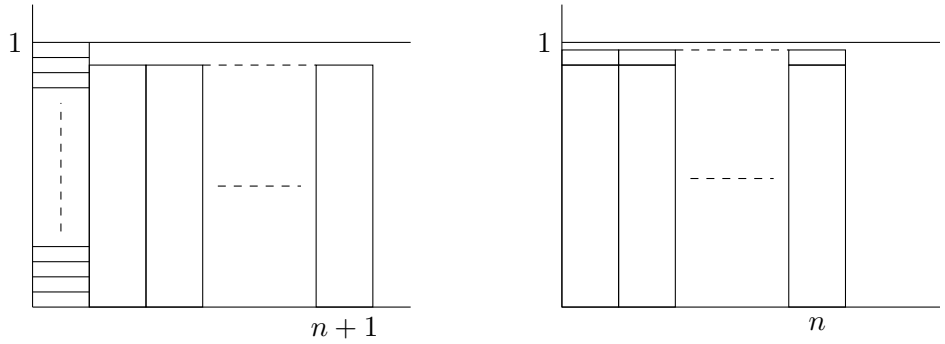


Figure 10.1: Lexmax-solution (left) and optimal solution (right) when minimizing the number of bins.

Calculating the value of V for both solutions, the first solution gives

$$V = n \cdot \frac{3}{2n} = \frac{3}{2}.$$

The optimal solution gives

$$V = n \cdot \left(1 - \frac{1}{2n}\right) = n - \frac{1}{2},$$

which is better with a number of analysts $n \geq 3$ and therefore also maximizing with respect to V gives the optimal solution in this case.

Bibliography

[1] E.H.L. Aarts and J.K. Lenstra. *Local search in combinatorial optimization.* Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1997.

[2] P. Brucker and S. Knust. *Complex Scheduling.* Springer, 2012.

[3] F. Glover. Tabu searchpart i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[4] F. Glover. Tabu searchpart ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[5] J.H. Holland. Adaptation in natural and artificial systems. *University of Michigan, Ann Arbor, MI*, 1975.

[6] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[7] J.Y-T. Leung. *Handbook of Scheduling.* Chapman and Hall/CRC, 2004.

[8] J.J. Schneider and M. Puchta. Investigation of acceptance simulated annealing a simplified approach to adaptive cooling schedules. *Physica A: Statistical Mechanics and its Applications*, 389(24):5822 – 5831, 2010.

[9] H.Y. Tarawneh, M. Ayob, and Z. Ahmad. Simulated annealing with dynamic initial temperatures for university course timetable problem. *Journal of Engineering and Applied Sciences*, 8(2):58–63, 2013.