MASTER’S THESIS

A GENERIC TASK PLANNING TOOLKIT

by

F.S. Bogerd BSc

Supervisors:

dr.ir. J.J.D. Aerts (CQM)
dr. K.J.M. Huisman (CQM)
dr.ir. C.A.J. Hurkens (TU/e)

Eindhoven, July 13, 2007
Abstract

The framework of a toolkit for solving scheduling problems is presented. The ideas for the toolkit come from four different scheduling cases. The toolkit consists of several modules, each with a different function. The computation module contains the implementation of three algorithmic approaches to construct a schedule, as well as a Graphical User Interface (GUI) Gantt chart object. The performance of the solvers is compared and conclusions are given about the options and limits of the toolkit.
Preface

This thesis describes my graduation project for my study of Industrial and Applied Mathematics at Eindhoven University of Technology (TU/e). The project, building a general toolkit for solving scheduling problems, took place at CQM. For eight months I worked there on my thesis in the planning group.

I want to use this opportunity to thank my supervisors from CQM, Joep Aerts and Kuno Huisman, who took time out of their busy schedule to keep me on track. I also appreciate their thorough scrutiny of my report and all the tips and suggestions they gave me these eight months. Also I would like to thank my supervisor from Eindhoven University, Cor Hurkens, for all the insight in solving optimization problems I got from our weekly meetings. I would also like to thank all the people at CQM for their patience in helping me to let AIMMS do what I wanted it to do.

Finally I would like to thank my girlfriend Xiaoting Yu, who did a very good job in keeping me motivated, always helped me focus on the important things, while keeping me well fed with her delicious Chinese dinners.

Finbar S. Bogerd
Eindhoven, July 13, 2007
Contents

Preface iii

1 Introduction 1
   1.1 MSc assignment description 2
   1.2 Outline of the report 3

2 Competitive software 5
   2.1 Initial results 6
   2.2 A closer look at Harmony 6

3 Planning cases 9
   3.1 Case 1: Workforce planning for an administrative company 10
   3.2 Case 2: Workforce planning for a printing company 21
   3.3 Case 3: Student assignment for the ID faculty 24
   3.4 Case 4: Mourik Services 28

4 Lagrange and flow 33
   4.1 Introduction to Lagrange relaxation 33
   4.2 Lagrange Relaxation 34
   4.3 Linking the IP model to a min cost max flow 36
   4.4 The relaxed extended planning problem 38
Chapter 1

Introduction

Scheduling and planning is a problem that many companies have to deal with. It may concern assigning tasks to employees or making a machine operating schedule.

Large companies who deal with huge instances of such scheduling and planning problems often have expensive and powerful Advanced Planning and Scheduling (APS) solutions to solve these problems efficiently.

In the small to medium sized business\(^1\) the available budgets are limited and therefore most APS solutions are out of reach. Furthermore, those APS solutions are often an overkill for the scope of the problems. The planning in such companies is mainly done by hand with additional use of e.g. Excel scripts.

One of the main reasons for choosing Excel is the availability, since almost every company has Microsoft Office installed by default, so there are no extra costs for software licences and hence the costs for planning in Excel are low.

The main questions for this master’s thesis are:

- Is it possible to create a custom made planning solution at low cost price?
- What is needed to create such an application?
- What range of problems can we tackle?

\(^1\)MKB or Midden en Klein Bedrijf in Dutch.
1.1 MSc assignment description

The definition we use here for scheduling and planning is assigning resources to tasks. For every task it is given what resources are needed and how much time of each resource is needed. For every resource the availability is given.

As an example consider the following case. Further details of this case can be found in Section 3.2 where this case is reviewed and implemented.

Given is the production plan of a big printing company. This planning is entirely based on the availability of printing machines and not on the availability of employees. From the machine planning the workforce demand can be derived by looking for each moment in time what printing machines are working and how many employees are needed to operate each machine. For every employee is it known when he is available and what his competences are, i.e. which tasks he can perform. The question is to find a personnel planning, given the production plan, such that all tasks in this plan can be performed.

In this way a large number of more or less identical relatively simple planning problems can be defined.

The planning problems are studied in the following three directions: algorithmic, technical, and market related.

Algorithmic questions

A number of planning problems can be solved by using time discretization and flow or matching algorithms. There are limits on what can be solved using this approach. The following questions are addressed:

- What are the prerequisites for a planning problem so that we can solve it using flow or matching algorithms?

- Is it possible to extend the flow algorithm with pre- and postprocessing to solve a wider selection of problems?

- What algorithms are available when flow or matching algorithms fail?
Technical questions

The resulting application will have the form of a toolkit with several components to perform different tasks. Components could be things like algorithms or interface pages e.g. a Gantt chart. This toolkit approach leads to the following questions:

- Is it possible to develop standard components in AIMMS such that building a custom made solution will be faster and hence cheaper?
- Is it possible to export the produced planning to e.g. Excel such that it can be easily distributed.
- Is it possible to make these solutions web-based?

Market questions

Besides technical and algorithmic questions also some market research has been performed as a part of this project. This involves both an analysis of the software that competitors offer and also finding customers, interested market segments, and their requirements for a solution.

Three cases described in this thesis come from CQM projects or customers and one problem comes from TU/e.

1.2 Outline of the report

In Chapter 2 an overview is given of the software that competitors offer. In Chapter 3 four planning cases used in this thesis are described. For these cases a problem description is presented as well as a flow model and a Mixed Integer Programming (MIP) model. Chapter 4 presents a solution method based on Lagrange relaxation combined with the flow solving library. In Chapter 5 four different algorithmic methods for scheduling problems are compared. Chapter 6 presents the framework of the toolkit and offers a detailed description of the different modules that together form the toolkit. In Chapter 7 an overview of the results is given as well as recommendations for further research.
Chapter 2

Competitive software

Part of this research is to find out what planning software is available in the market. In order to get a clear comparison between the different competitive software packages and companies who offer planning solutions a list of standard items is used. From all companies that offer planning software packages the following information\textsuperscript{1} is gathered:

1. Website (url).
2. Size of the company.
3. Key customers of the company.
4. Customer focus (market segments).
5. Name of software package.
6. Is planning the key focus of the software.
7. Extra functions offered by package.
8. Costs of package (structure and values).
9. Web functionality for employees and planners.

\textsuperscript{1}All the information used in this overview is found on the website of the company or in documents offered by the company on their website.
2.1 Initial results

In Appendix A the full result of the search is given. In this search solutions from the following companies were considered (between brackets is the name of the software package):

- Adecco (E-tools)
- Arpa Interbouw
- Bit by Bit information systems (IRIS-Planning)
- Invision Software (AutoScheduler)
- Microsoft (Navision Axapta)
- Mijnrooster
- OM Partners (OMP Supply Chain Suite)
- Ortec (Harmony)
- Paralax (Rostar CAS)
- Primavera
- Quintiq
- Sap (APO)
- SP-Expert: Intus
- SP-Expert: Planning IT
- SP-Expert: Planpoint (Rooster direct pakket)
- SP-Expert: Spen
- Symagic (De Symagic workforce management oplossing)
- Synusflex
- Unit4agresso (Unit 4 Multivers Productie)

Note that four different companies offer a planning solution based on the SP-Expert framework.

All the packages that have been considered in this research are complete suites. They include many additional features like links with hour registration databases and the performance of checks on legal requirements. These suites are mainly designed to assist manual planning of resources.

Even if planning is the key focus of the software, automated planning/scheduling is not. If automated scheduling is offered as a feature then no guarantees are made about the quality of the solution. Only one program (the SP-Expert solution from Planning IT) mentions the type of algorithm (genetic algorithm) used by the automated planning module.

2.2 A closer look at Harmony

As a second step one package is analyzed in more detail. In this section a closer look is taken at Harmony and its usefulness within the scope of this master’s thesis is described.
In order to get a good insight at the use of Harmony a meeting was arranged with Mrs. Oskam, who is responsible for the functional maintenance of Harmony at a hospital [Harmony, 2007].

Harmony, the suite offered by Ortec, has a wide scope of possibilities. The instance described here is implemented in a hospital environment. This hospital has chosen Harmony for its administrative options. All the hour registrations and employee salary data is exported from Harmony.

The intention was to also use Harmony for automatic planning. In practice the scheduling problem offered turned out to be too complex to solve. So Harmony is used in computer aided scheduling as a decision support tool. Which means that there is still a person manually creating the personnel planning, but Harmony will check for legal and preferential constraints and gives warnings if they are exceeded.

The interface of Harmony has a look and feel that is very similar to Microsoft Excel. This improves the usage for most planners since they have usually no or limited experience with advanced planning packages, while Excel is a commonly used program.

The hospital in question has about 5000 to 6000 employees in about 400 groups, where one employee can be in more than one group. Each group has it’s own planning and planner. These are the main reasons why Harmony is implemented as a client - server system, where each of the planners can log on to.

On many occasions the capacity of the three servers employed at the hospital is a limiting factor in the use of Harmony and the planning module. Although there is a wish for automatic generated planning (especially by the larger groups) until this day all the planning is done by hand, aided by Harmony.

The hospital does not use any of the web functions Harmony offers. All communication with the planned personnel is done outside Harmony and the planners can only log in over the intranet.

In conclusion: Harmony offers support for planners and has the functionality to solve scheduling problems. In this case the administrative functions and constraint checking abilities of Harmony are used to aid in the creation of a valid planning.
Chapter 3

Planning cases

In this chapter three planning cases are considered. The implementation of these three cases is used as the starting point for the generic AIMMS planning toolkit.

The three cases that are considered are:

- In Section 3.1 workforce planning for an administrative company.
- In Section 3.2 workforce planning for a printing company.
- In Section 3.3 student assignment for the faculty of Industrial Design at the TU/e.

For each of the cases a problem description is presented. Then the problem is written as a flow model and an overview of the implementation in AIMMS is given.

Definition. Throughout this report the following parameters are used:

- $\mathcal{T}$ Timeslots (with index $t$)
- $\mathcal{E}$ Employees (with index $e$)
- $\mathcal{J}$ Types of jobs / tasks (with index $j$)
- $\mathcal{I} \subset \mathcal{J}$ Types of jobs / tasks (with index $i$) that have a follow up appointment
- $\mathcal{I}^* \subset \mathcal{J}$ Types of jobs / tasks (with index $i^*$) that are follow up appointments
- $\mathcal{M}$ Machines (with index $m$)
- $A_{et}$ Employee availability per time unit $\in \{0, 1\}$
- $C_{ej}$ Employee competence per job $\in \{0, 1\}$
\( F_{jt} \) Job frequency / requirement per time unit \( \in \mathbb{N} \)

\( G_{mt} \) Machine requirement per time unit \( \in \mathbb{N} \)

\( H_{jm} \) Job requirement per machine \( \in \mathbb{N} \)

\( v \) Vertices / nodes

\( v_{so} \) Source vertex

\( v_{si} \) Sink vertex

\( b \) Maximum number of timeslots between a task and its follow up task

\( x, y \) Decision variables

We will use \(|\mathcal{E}|\) to denote the cardinality of the set \( \mathcal{E} \).

### 3.1 Case 1: Workforce planning for an administrative company

First the problem is discussed, then a network flow model is introduced and an implementation in AIMMS is given. Then more complex conditions for this case are added as extensions.

#### 3.1.1 Problem description

At an administrative company there are tasks that need to be assigned to employees. For each task it is given which employees have the right competence to carry out that task. The problem is to find a schedule such that every task is carried out by a competent person.

**Specification**

The problem consists of the set employees \( \mathcal{E} \), the set of task-types \( \mathcal{J} \) and the set of timeslots \( \mathcal{T} \). Every workday consists of two timeslots, one in the morning and one in the afternoon. A week consists of 10 timeslots. For each timeslot it is specified how many of each task-type need to be carried out and which employees are present during that timeslot.

The number of times task-type \( j \) needs to be carried out on timeslot \( t \) is given by the job demand matrix \( F \). The availability of employee \( e \) on timeslot \( t \) is given by the availability matrix \( A \). The competence of employee \( e \) for task-type \( j \) is given by the competence matrix \( C \).

In the final schedule for each time unit \( t \), each task needs to be assigned to a competent employee.
Follow up appointments

A difficulty in this case is that some task-types require a follow up task that has to take place within $b$ timeslots after the associated first task. This follow up task will also take one timeslot, and needs to be carried out by the same employee who did the associated first task.

The starting model is a flow model that does not include task-types that require a follow up task. The case including follow up tasks is addressed in Section 3.1.5.

3.1.2 Flow model

In this section a model is given to solve the problem based on flow optimization. There is no dependence or interaction between different timeslots so each timeslot can be seen as a stand alone optimization problem.

For each timeslot $t$ the flow model is constructed and solved.

In Figure 3.1 a graphical representation is depicted of the flow model. The flow model is a graph. In this graph there are two main types of vertices. Employee vertices $\tilde{v}_e$ and task vertices $\hat{v}_j$. Two additional vertices, a source vertex $v_{so}$ and a sink vertex $v_{si}$ are added.

![Figure 3.1: The flow model of an employee-task problem.](image)

The directed edges in this graph consist of

- Edges $(v_{so}, \tilde{v}_e)$ with capacity $A_{et}$
- Edges $(\tilde{v}_e, \hat{v}_j)$ with capacity $C_{ej}$
- Edges $(\hat{v}_j, v_{si})$ with capacity $F_{jt}$
Then a maximum flow algorithm is applied to this graph to find a flow from \( v_{so} \) to \( v_{si} \) with a value of \( \sum_{j \in \mathcal{J}} F_{jt} \).

Theorem 4.2 from [Nemhauser and Wolsey, 1988] states:

*If all the arc capacities are integer-valued, then there is a maximum flow \( x \in \mathbb{Z}_+^n \).*

Here all arc capacities from employees to tasks have a maximum capacity of either 0 or 1. The resulting flow over these arcs will be integer and thus either 0 or 1. An arc with flow 1 means that the employee is matched to that type of task in the schedule. Hence, a feasible solution to the flow problem corresponds to a feasible schedule.

### 3.1.3 MIP formulation

This case can also be formulated as a Mixed Integer Program (MIP). For this formulation the variable \( x_{ejt} \) is introduced, where \( x_{ejt} = 1 \) when employee \( e \) performs a task of type \( j \) on timeslot \( t \) and zero otherwise.

Let \( \mathcal{E} \) with index \( e \) be the set of employees, \( \mathcal{J} \) with index \( j \) be the set of tasks and \( \mathcal{T} \) with index \( t \) be the set of timeslots for which a planning has to be found. Let \( F_{jt} \) be the number of times task-type \( j \) has to be done in timeslot \( t \). Let \( A_{et} = 1 \) if employee \( e \) is available at timeslot \( t \) and let \( A_{et} = 0 \) if employee \( e \) is not available at timeslot \( t \). Let \( C_{ej} = 1 \) if employee \( e \) has the competence to execute task \( j \) and let \( C_{ej} = 0 \) otherwise.

The restrictions for this model are described below.

Task \( j \) on timeslot \( t \) must be performed exactly \( F_{jt} \) times,

\[
\sum_{e \in \mathcal{E}} x_{ejt} = F_{jt}.
\]

Employee \( e \) can only perform task \( j \) if he is available in timeslot \( t \),

\[
x_{ejt} \leq A_{et}.
\]

Employee \( e \) can only perform task \( j \) if he is competent to do so,

\[
x_{ejt} \leq C_{ej}.
\]

Employee \( e \) can only perform task \( j \) completely or not at all,

\[
x_{ejt} \in \{0, 1\}.
\]
The full MIP formulation becomes:

$$\begin{align*}
\text{Max} & \quad \sum_{e \in E, j \in J, t \in T} x_{ejt} \\
\text{s.t.} & \quad \sum_{e \in E} x_{ejt} = F_{jt}, \quad \forall j \in J, t \in T, \\
& \quad \sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in E, t \in T, \\
& \quad x_{ejt} \leq C_{ej}, \quad \forall e \in E, j \in J, t \in T, \\
& \quad x_{ejt} \in \{0, 1\}, \quad \forall e \in E, j \in J, t \in T.
\end{align*}$$

(3.1)

3.1.4 AIMMS implementation

The basic flow model is implemented in AIMMS. The implementation has been done in the following steps:

1. Read the problem data from the Microsoft Excel files.
2. Construct the AIMMS model in terms of employees and tasks.
3. Convert the AIMMS model to the graph formulation.
4. Execute the Goldberg flow library from AIMMS.
5. Convert the result from Goldberg to Excel output.

Read from Excel

The problem input is given in an Excel file. On the first sheet a table is given that denotes who is not present at certain timeslots. The table contains a 1 if the employee is not present on that timeslot and a 0 if the employee is present. The second sheet describes which employee is competent for what task. The table contains a 1 if the employee is competent to do the task and a 0 otherwise. The third table gives how many of each task have to be done on each timeslot. In AIMMS the Excel file is read by use of the following code:

```aimms
ExcelSetActiveSheet(Workbook,tab_Afwezig); /* 1 */
ExcelRetrieveSet(Workbook,s_dagen,"Ad"); /* 2 */
ExcelRetrieveSet(Workbook,s_namen,"An"); /* 3 */
ExcelRetrieveTable(Workbook,p_Afwezig,"Aa","An","Ad"); /* 4 */
ExcelSetActiveSheet(Workbook,tab_Bevoegd); /* 5 */
ExcelRetrieveSet(Workbook,s_taken,"Bt"); /* 6 */
ExcelRetrieveTable(Workbook,p_Bevoegd,"Bb","Bn","bt"); /* 7 */
```
ExcelSetActiveSheet(Workbook,tab_Aantaltaken); /* 8 */
ExcelRetrieveTable(Workbook,p_Aantaltaken,"Ta","Tt","Td"); /* 9 */

Line 1, 5, and 8 open the right excel sheet, lines 2, 3, and 6 read the timeslot, employee names and task type names. Lines 4, 7, and 9 read the availability, competence and number of tasks from the excel file. The two letter ranges (e.g. “Ad”, “Bb”) are sets of cells that are defined in the excel file.

Construct the AIMMS model

In AIMMS a model is created. This model contains the sets timeslots, names and tasks. Also this model contains the parameters present (timeslot, name), competent (name, task) and numberofjobs (task, timeslot). In these parameters a 0 means a person is not available at that timeslot or not competent to perform the task. Moreover, a 1 means the person is available or is competent to do the task. The numberofjobs parameter denotes how many times a task has to be performed in that timeslot.

Convert the AIMMS model to a flow model

In order to use the Goldberg flow code the model needs to be defined as a flow model. Hence the problem must be written in terms of nodes and arcs. In the simple case, the planning problem of each timeslot can be seen as a separate flow problem. Hence, the AIMMS flow models only one timeslot. For every timeslot this model is used and the Goldberg code is applied.

In AIMMS two sets are created: nodes and arcs. The set nodes consists of the sets names and tasks The nodes source and sink are added to the set of nodes. For every arc a parameter is used that describes the begin and endnode. Also parameters are specified that define the capacity of the arc and the cost per unit of flow through the arc. Finally a parameter is introduced that gives the supply of each node. This parameter denotes how much flow is created or absorbed in each node. In this case arcs are created:

- From the source to a name if the person is available \((A_{et} = 1)\) with capacity 1.
- From a name to a task if that person is capable to perform the task \((C_{ej} = 1)\) also with capacity 1.
- From a task to the sink with capacity \(F_{jd}\) the number of times the task needs to be performed.
The supply of all nodes is zero except for the source and the sink. In the source a flow of \( z \) is created and in the sink a flow of \( z \) is absorbed. Where \( z \) denotes the total number of tasks that need to be performed.

A source-sink arc can be added with high capacity and costs to make sure the Goldberg code gives a partially feasible solution when a completely feasible solution does not exist.

**Execute the Goldberg library**

Goldberg has produced a library\(^1\) that contains an optimal flow algorithm. An existing AIMMS link with the Goldberg library created by CQM is used. For more information about the Goldberg algorithm see [Goldberg, 1997], see [Teeuwen and Lamers, 2004] for more information about the AIMMS link.

The input of the Goldberg library is a graph \( G(V,U) \) with edge costs, minimum and maximum edge capacities and excess at each vertex. All these parameters must be integer valued. The minimum edge capacity must be lower or equal to the maximum edge capacity and the sum over all vertices of the excess must be zero. The output of the Goldberg library is a parameter that gives for each arc the amount of flow over this arc in the min cost flow. In this simple case where all arc costs are set to 1, the output will be a feasible solution to the original problem.

**Export the results to Excel**

Once the Goldberg library has produced the schedule, the schedule is written to Microsoft Excel by means of the following code:

```plaintext
ExcelSetActiveSheet(Workbook, tab_Schema); /* 1 */
ExcelClearRange(Workbook,"ss"); /* 2 */
ExcelClearRange(Workbook,"sn"); /* 3 */
ExcelClearRange(Workbook,"sd"); /* 4 */
ExcelAssignSet(Workbook,s_namen,"sn"); /* 5 */
ExcelAssignSet(Workbook,s_dagen,"sd"); /* 6 */
ExcelAssignTable(Workbook,ep_output_rooster,"ss","sn","sd",rowmode:1, columnmode:1); /* 7 */
ExcelSaveWorkbook(Workbook); /* 8 */
```

Line 1 opens the right excel sheet, lines 2 till 4 clears the datafields, line 5 writes the set of names to the range “sn”, line 6 writes the set of timeslots to the range “sd”, line 7

\(^1\)The library can be downloaded from [http://www.avglab.com/andrew/soft.html](http://www.avglab.com/andrew/soft.html)
writes the schedule to in the range specified by “sd” and finally line 8 saves and closes the workbook. The ranges (“sn”, “sd” and “ss”) are sets of cells that are defined in the Excel output file.

3.1.5 Extensions of the case

In this section two additional features to the case are presented. The first one is variation in the output schedule. The flow algorithm is a static algorithm, given the same input instance it always produces the same output. The result of this is that if a week has the same parameters \((A_{et}, F_{jt})\) as a previous week, the same schedule as the previous week is produced for this week. Variation is essential for most employees so it is desired to get a different (feasible) schedule each week.

The second extension is the introduction of the follow up appointments as explained previously. In this extension there are certain task types that require a follow up task, that needs to be carried out by the employee who performed the first task within a predefined number \((b)\) of timeslots.

Variation in the schedule

With the same input the flow algorithm always produces exactly the same output. In order to change this a small change is made to the input of the flow algorithm. Every timeslot the order of the set of nodes is changed. This means that every timeslot the flow algorithm has a completely different ordering of nodes in the graph, and thus produces with high probability a different valid solution (if one exists).

If the algorithm is run more times to plan the same week, every run a different schedule is produced. The randomization of the order of the nodes is done with the AIMMS command order by. A parameter is created for every node in the flowgraph. Before the flowgraph is given to the algorithm this parameter is assigned a uniformly distributed value between 0 and 1. Then the nodes are sorted according to this parameter and hence a new order is created for the set of nodes.

Follow up appointments

Several task-types from the task list are special in the sense that they have a follow up appointment, these special task types are called leading tasks. The rules for a follow up appointment are that they need to be carried out within a certain number of timeslots from the their leading task and that they need to be carried out by the same employee who performed the leading task. The follow up appointments make the problem difficult
in a sense that the max flow algorithm can no longer be applied. A person who performs a follow up task can no longer perform another task on that timeslot. This restriction cannot be captured within an arc node graph, so something new needs to be constructed. A first solution is to start with scheduling the follow up appointments by means of greedy assignment. Assigning leading tasks and their follow up task to available, competent employees and then find a schedule (using flow) for the still available employees and the simple tasks that still need to be carried out. A task is called simple if it is neither a leading nor a follow up task. Later other solutions are proposed.

The heuristic applied in the first solution is that of greedy matching followed by flow. The tasks with follow up, and their follow up tasks are planned greedy, so the first employee that can perform the task and the follow up task gets them assigned. The remaining simple tasks are planned using the flow module given in the basic case solution. To even the workload and introduce variation for all the employees, the order in which the employees are searched is changed every timeslot.

The main problem with greedy matching is that it can make the resulting flow problem infeasible. This can happen because the people assigned to follow up tasks cannot be assigned to other tasks. Due to the fact that every task can only be performed by a subset of the employees, it can happen that there are tasks that cannot be performed anymore due to a lack of competent available employees.

With the data from the administrative company this happens rarely. The quick fix applied to this problem is to empty the schedule and restart the greedy matching procedure whenever the flow problem becomes infeasible. This solution fails when the probability that the flow problem becomes infeasible increases. In that case alternative methods like MIP or Lagrange relaxation can be applied. These methods are discussed in the next subsection and the next chapter respectively.

3.1.6 MIP formulation of follow up appointments

The follow up appointments can also be modeled as constraints in the MIP formulation (3.1) of the case.

Below \( t, t_1 \) and \( t_2 \) are indices of the set of time units \( T \) and \( i \) is the index of the set of leading task types \( I \). Let \( i^* \) be the follow up task defined for job \( i \), where \( I^* \) is the set of follow up tasks. A task-type cannot be both in \( I \) and \( I^* \). Let \( b \) be the maximum number of days between a leading task and its follow up task.

The added constraints must model the system requirement that each type \( i \) task has a type \( i^* \) follow up task within \( b \) timeslots. After the MIP formulation a validation of these
constraints is given. The constraints are derived from the idea that for every interval \((t_1, t_2)\) with \(t_1 \leq t_2\) the following must hold:

- The number of type \(i\) tasks in the interval \((t_1, t_2)\) must be less or equal to the number of type \(i^*\) tasks in the interval \((t_1 + 1, t_2 + b)\), since each task of type \(i\) in the interval \((t_1, t_2)\) must have a unique matching task in the interval \((t_1 + 1, t_2 + b)\).
- The number of type \(i^*\) tasks in the interval \((t_1, t_2)\) must be less or equal to the number of type \(i\) tasks in the interval \((t_1 - b, t_2 - 1)\), since each task of type \(i^*\) in the interval \((t_1, t_2)\) must have a unique matching task in the interval \((t_1 - b, t_2 - 1)\).

This results in an enormous amount of constraints. For example for 3 types of follow up tasks, 20 employees and 30 timeslots there are:

\[
2 \times 3 \times 20 \times \binom{30}{2} = 52200
\]

constraints.

As proved in the next paragraph, this number can be decreased by only looking at the intervals that start at the first timeslot or end at the last timeslot. In the same example this gives only:

\[
4 \times 3 \times 20 \times 30 = 7200
\]

constraints. The resulting constraints still model the requirement of follow up tasks.

The resulting MIP formulation is as follows:

\[
\begin{align*}
\text{Max} & \quad \sum_{e \in E, j \in J, t \in T} x_{ejt} \\
\text{s.t.} & \quad \sum_{e \in E} x_{ejt} \leq F_{jt}, \quad \forall j \in J \setminus I^*, t \in T, \\
& \quad \sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in E, t \in T, \\
& \quad x_{ejt} \leq C_{ej}, \quad \forall e \in E, j \in J, t \in T, \\
& \quad x_{ejt} \in \{0, 1\}, \quad \forall e \in E, j \in J, t \in T,
\end{align*}
\]

(3.2)

In Chapter 4 Lagrange relaxation in combination with flow is used to solve this MIP.
Validation of the MIP definition

If a person has a task of type $i$ on timeslot $t$ then within $b$ timeslots after time $t$ he/she must have a follow up task (type $i^*$). Note that a type $i^*$ task can never occur by itself but only as a result of a type $i$ task and a person can only have one task each timeslot.

Without loss of generality say there:

- is only one task type that requires a follow up task. Automatically there is only one task type in the set $I^*$.
- are no task types that do not have a follow up tasks.
- is only one employee available.

**Theorem 1.** Let $x_t = 1$ if the person is assigned a task that requires a follow up task on timeslot $t$ and 0 otherwise. Let $y_t = 1$ if the person has a follow up task on timeslot $t$ and 0 otherwise. Then the problem, with one person and two task types, can be modeled by the following IP:

\[x_t + y_t \leq 1, \quad \forall t \in T,\]  \hspace{1cm} (3.3)

\[\sum_{t_2 \leq t_1} x_{t_2} \leq \sum_{t_2 \leq t_1 + b} y_{t_2}, \quad \forall t_1 \in T,\]  \hspace{1cm} (3.4)

\[\sum_{t_2 \geq t_1} x_{t_2} \leq \sum_{t_2 \geq t_1 + 1} y_{t_2}, \quad \forall t_1 \in T,\]  \hspace{1cm} (3.5)

\[\sum_{t_2 \leq t_1} y_{t_2} \leq \sum_{t_2 \leq t_1 - 1} x_{t_2}, \quad \forall t_1 \in T,\]  \hspace{1cm} (3.6)

\[\sum_{t_2 \geq t_1} y_{t_2} \leq \sum_{t_2 \geq t_1 - b} x_{t_2}, \quad \forall t_1 \in T.\]  \hspace{1cm} (3.7)

**Lemma 2.** The constraints given in theorem 1 imply the constraint:

\[\sum_{t \in T} x_t = \sum_{t \in T} y_t.\]  

**Proof.** Let $t = 0$ be the first timeslot and let $t = T$ be the last timeslot.

Constraint (3.5) with $t_1 = T$ gives $x_T = 0$.

Constraint (3.6) with $t_1 = T$ then gives that $\sum_{t \in T} y_t \leq \sum_{t \in T} x_t - x_T = \sum_{t_2 \in T} x_{t_2}$.

Constraint (3.4) with $t = T$ gives $\sum_{t \in T} x_t \leq \sum_{t \in T} y_t$ and the lemma follows. \hfill $\Box$

A task matching is a pairing of type $i$ and type $i^*$ tasks such that:

- Every type $i^*$ task is matched to exactly one type $i$ task.
• Every type $i$ task is matched to exactly one type $i^*$ task.
• Every type $i^*$ task is no more than $b$ timeslots later than the matched type $i$ task.
• Every type $i^*$ task is later than the matched type $i$ task.
• If task $p$ is matched with task $q$, then task $q$ is matched to task $p$.

**Lemma 3.** If conditions (3.4) to (3.7) hold, then a task matching as described above is possible.

**Proof.** Assume (3.4) to (3.7) hold. Lemma 2 gives that the number of type $i$ tasks is equal to the number of type $i^*$ tasks. So the remaining part of this proof is that there exists a matching where the tasks are all within appropriate distance.

![Figure 3.2: Different intervals of the timeslots.](image)

Assume no matching can be found. This means that there has to be:

- Case 1: an interval from $t_1$ to $t_2$ (interval $S_2$ in Figure 3.2) that has more type $i$ tasks than there are type $i^*$ tasks in $t_1 + 1$ to $t_2 + b$ (interval $S_5$ in Figure 3.2), or
- Case 2: an interval from $t_1$ to $t_2$ that has more type $i^*$ tasks than there are type $i$ tasks in $t_1 - b$ to $t_2 - 1$.

In case 1 the interval $S_2$ contains more type $i$ tasks than there are type $i^*$ tasks in $S_5$. Measured over the whole time space the number of type $i$ jobs is equal to the number of type $i^*$ jobs. So at least one of the following statements is true:

1. the interval $S_1$ contains less type $i$ tasks than there are type $i^*$ tasks in $S_4$,
2. the interval $S_3$ contains less type $i$ tasks than there are type $i^*$ tasks in $S_6$,

When statement 1 is true the interval $t_1$ to $t_2$ can be extended to the interval from $t_1$ to $t_T$ (interval $S_2 + S_3$ in Figure 3.2). Then the number of tasks $i$ in the interval $t_1$ to $t_T$ is more than then number of $i^*$ tasks on the interval $t_1 + 1$ to $t_T$. This is in contradiction to the
assumption that \((3.5)\) holds. When statement 2 is true the interval \(t_1\) to \(t_2\) can be extended to the interval from \(t_0\) to \(t_2\) (interval \(S_1 + S_2\) in Figure 3.2). Then the number of tasks \(i\) in the interval \(t_0\) to \(t_2\) is more than then number of \(i^*\) tasks on the interval \(t_0\) to \(t_2 + b\). This is in contradiction to the assumption that \((3.4)\) holds. Hence a matching can be found.

The proof for case 2 is equivalent.

**Lemma 4.** If there is an task matching as described above then conditions \((3.4)\) to \((3.7)\) hold.

**Proof.** Assume there is an task matching as described above and that conditions \((3.4)\) to \((3.7)\) do not hold.

Since conditions \((3.4)\) to \((3.7)\) do not hold there is either:

1. an interval from \(t_1\) to \(t_2\) that has less type \(i\) tasks than there are type \(i^*\) tasks in \(t_1 + 1\) to \(t_2 + b\), or
2. an interval from \(t_1\) to \(t_2\) that has less type \(i^*\) tasks than there are type \(i\) tasks in \(t_1 - b\) to \(t_2 - 1\).

In each of those cases an task matching between the type \(i\) and type \(i^*\) tasks is not possible.

- In situation 1 it is not possible for every task of type \(i\) in the interval \([t_1, t_2]\) to find a matching type \(i^*\) task. All these matching type \(i^*\) tasks need to come from the interval \([t_1 + 1, t_2 + b]\) and there are not enough type \(i^*\) tasks in this interval.
- In situation 2 it is not possible for every task of type \(i^*\) in the interval \([t_1, t_2]\) to find a matching type \(i\) task. All these matching type \(i\) tasks need to come from the interval \([t_1 - b, t_2 - 1]\) and there are not enough type \(i\) tasks in this interval.

So the lemma holds.

**Proof.** (of Theorem 3.1) Since a solution to our problem is the same as a valid task matching, Lemma 3 and Lemma 4 combined give the proof.

### 3.2 Case 2: Workforce planning for a printing company

This case is about a big printing company. In this printing company there are different machines present and a job schedule is made based on the availability of these machines.
The problem of this case is: given a schedule of the machines (which machine works at what time), find the right personnel to operate the machines such that the machine schedule can be executed.

A difference with the previous case is that now one machine could have more tasks attached to it that need to be performed in order to let it operate. Every one of these tasks needs a different competence.

3.2.1 Problem description

The problem consists of the set employees $E$, the set tasks $J$, the set of machines $M$ and the set of time units (in this case timeslots) $T$. Every day consists of four slots, two in the morning and two in the afternoon, therefore a week consists of 20 timeslots. For each timeslot it is specified what machines operate and which employees are present during that timeslot.

Whether machine $m$ needs to operate on timeslot $t$ is given by the machine demand matrix $G$. The availability of employee $e$ on timeslot $t$ is given by the availability matrix $A$. The competence of employee $e$ for task $j$ is given by the competence matrix $C$. The tasks $j$ needed to operate machine $m$ are given by the job requirement matrix $H$.

In the final schedule for each time unit, all tasks need to be assigned to a competent employee.

3.2.2 Flow model

This planning problem is discretized in time, so every timeslot can be considered as a stand alone problem. In order to model each timeslot $t$ as a flow problem in a graph we need to build up the nodes and arcs of the graph. The set of nodes consists of employee nodes ($\tilde{v}$), task nodes ($\hat{v}$), machine nodes ($v$) a source node $v_{so}$ and a sink node $v_{si}$.

Based on the model variables edges are created between the different arcs:

- edges $(v_s, \tilde{v}_e)$ with capacity $A_{et}$,
- edges $(\tilde{v}_e, \hat{v}_j)$ with capacity $C_{ej}$,
- edges $(\hat{v}_j, \tilde{v}_k)$ with capacity $H_{jm}$,
- edges $(\tilde{v}_k, v_t)$ with capacity $G_{mt}$.

The problem is to find a flow of value $\sum_{m \in M} G_{mt}$ from node $v_{so}$ to node $v_{si}$ in the graph.
Every such solution defines a personnel planning for timeslot $t$. Since an employee is either assigned to a task, the arc $(\hat{e}, \hat{j})$ has flow 1, or not assigned to a task, the arc has flow 0.

### 3.2.3 MIP formulation

This problem can also be modeled as an MIP. The variable $x_{ejt}$ is 1 if employee $e$ performs task $j$ on timeslot $t$ and zero otherwise. The following restriction are imposed:

If task $j$ is required for machine $m$ and machine $m$ is required to work on timeslot $t$ then this task must be performed by an employee, which gives:

$$\sum_{e \in E} x_{ejt} = \sum_{m \in M} H_{jm} G_{mt}, \quad \forall j \in J, t \in T,$$

Employee $e$ can only perform task $j$ on timeslot $t$ if the employee is available,

$$\sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in E, t \in T,$$

Employee $e$ can only perform task $j$ if he is competent to do so,

$$x_{ejt} \leq C_{ej}, \quad \forall e \in E, j \in J, t \in T,$$

Employee $e$ can only perform task $j$ completely or not at all,

$$x_{ejt} \in \{0, 1\}, \quad \forall e \in E, j \in J, t \in T.$$

the complete MIP problem description then becomes:

$$\text{Max} \quad \sum_{e \in E, j \in J, t \in T} x_{ejt}$$

s.t.

$$\sum_{e \in E} x_{ejt} = \sum_{m \in M} H_{jm} G_{mt}, \quad \forall j \in J, t \in T,$$

$$\sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in E, t \in T,$$

$$x_{ejt} \leq C_{ej}, \quad \forall e \in E, j \in J, t \in T,$$

$$x_{ejt} \in \{0, 1\}, \quad \forall e \in E, j \in J, t \in T.$$

(3.8)

### 3.2.4 AIMMS implementation

The printing company problem is also implemented in AIMMS. The main difference between this case and the case of Section 3.1 is the presence of an additional layer, namely the
machines layer. The matching between tasks and machines is fixed (if machine \( k \) is needed all tasks attached to it are needed), so the machines layer and tasks layer can be replaced by one layer. This one layer is identified by the task. The capacity of the edge between the task and the sink is 1 if the machine which is linked to the task is needed and a 0 otherwise.

The problem now becomes finding a maximum flow in a graph with two layers of nodes. This is already solved in the previous case and the AIMMS functions written in the previous section can be used. Hence, a feasible solution is easily found when it exist.

### 3.3 Case 3: Student assignment for the faculty of Industrial Design from the TU/e

#### 3.3.1 Problem description

At the faculty of Industrial Design (ID) at Eindhoven University of Technology students have to take a few elective courses each semester. The student picks these courses from a predefined set. Each course has a maximum number of students that can attend the course. A student sends in an ordered list of ten preferred courses and how much elective courses he/she wants to take. For the elective courses a number of time slots is reserved in the schedule. A student can have only one subject in each available timeslot.

In the past the planner at the faculty used a greedy algorithm to assign students to courses. Although this resulted in a few unhappy students there was nothing wrong. The number of students at the faculty of ID grew significantly and the greedy algorithm no longer gave acceptable results.

In this section the ID case is implemented as a flow problem. Furthermore, an AIMMS implementation of the case is given. In Subsection 3.3.5 some extensions of the case are discussed. This case has been taken from [v.d. Broek et al., 2006].

#### 3.3.2 Flow model

Although this problem substantially differs from the previous cases, the same parameter names are used where they denote a similar meaning. For this problem some new parameters are introduced as well.

Let \( P_e \) be the preferred number of elective courses student \( e \) wants to attend. Let \( R_{e,j} \) be the ranking that student \( e \) has given course \( j \). The ranking is 0 if a student did not put
a course on his list, the most desired course gets rank 1, the second most desired course gets a 2, the third most a 3 etc. Let $U_{jt}$ be 1 if course $j$ is given in timeslot $t$ and 0 otherwise. Let $C_{e j}$ be one if student $e$ has course $j$ on his preference list and zero otherwise. Let $S_j$ be the maximum number of students allowed in a course. Let $E$ with index $e$ be the set of students, let $T$ with index $t$ be the set of timeslots and let $J$ with index $j$ be the set of courses.

The resulting flow graph\textsuperscript{2} is given in Figure 3.3.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{flow-model.png}
\caption{The flow model of the ID elective courses problem.}
\end{figure}

There are three layers of nodes. Every arc goes from one layer to a higher layer. The arcs are layered as follows:
- student names,
- student - timeslot (a different node for each student - timeslot combination),
- courses.

There also is a source node $v_s$ as layer 0, and a sink node $v_t$ as layer 4.

The following names are used for the sets of nodes: $\tilde{v}$ for student nodes, $\hat{v}$ for student-timeslot nodes, $\overline{v}$ for the course nodes, $v_{so}$ for the source and $v_{si}$ for the sink.

Based on the model variables edges are created between the different arcs:
- edges $(v_{so}, \tilde{v}_e)$ with capacity $P_e$,
- edges $(\tilde{v}_e, \hat{v}_{et})$ with capacity 1,
- edges $(\hat{v}_{et}, \overline{v}_j)$ with capacity $C_{\text{max}}$.

\textsuperscript{2}The graph is taken from v.d. Brock et al., 2006.
• edges $(\bar{v}_{et}, \bar{v}_{j})$ with capacity $C_{ej}U_{jt}$ and costs $R_{ej}U_{jt}$,
• edges $(\bar{v}_c, v_s)$ with capacity $S_j$.

If no costs are specified for an arc then the arc has costs 1.

### 3.3.3 MIP formulation

The ID case can also be modelled as an MIP. Let the variables $x_{ej}$ be one if student $e$ gets course $j$ assigned and zero otherwise. The MIP problem is a two step problem, the first step is to find the maximum number of course-spots that can be assigned to students when every student has no more courses assigned to him that he or she desired.

The restrictions imposed are: Student $e$ only wants course $j$ if it is on his preference list,

$$x_{ej} \leq C_{ej}.$$

Student $e$ can have at most one course in timeslot $t$,

$$\sum_j x_{ej}U_{jt} \leq 1.$$

Course $j$ has maximum number of $S_j$ students,

$$\sum_e x_{ej} \leq S_j.$$

Student $e$ can have no more courses than preferred $P_e$,

$$\sum_j x_{ej} \leq P_e.$$

The full MIP formulation of the first step is as follows:

$$\begin{align*}
\text{Max} & \quad \sum_{e \in \mathcal{E}, j \in \mathcal{J}} x_{ej}m, \\
\text{s.t.} & \quad x_{ej} \leq C_{ej}, \quad \forall e \in \mathcal{E}, j \in \mathcal{J}, \\
& \quad \sum_j x_{ej}U_{jt} \leq 1, \quad \forall e \in \mathcal{E}, t \in \mathcal{T}, \\
& \quad \sum_j x_{ej} \leq S_j, \quad \forall j \in \mathcal{J}, \\
& \quad \sum_j x_{ej} \leq P_e, \quad \forall e \in \mathcal{E}, \\
& \quad x_{ej} \in \{0, 1\}, \quad \forall e \in \mathcal{E}, j \in \mathcal{J}. 
\end{align*}$$

(3.9)
Let $Y$ be maximum value \([3.9]\) attains.

The second step is make an assignment of $Y$ courses and making the sum of the rankings of the course-spots as low as possible. Compared to Step 1 we add the constraint that $Y$ courses must be assigned. Moreover the objective function is changed to take the course ranking into account.

Step 2:

$$\text{Min} \sum_{e \in E, j \in J} x_{ej} R_{ej},$$

s.t.

$$\sum_{e \in E, j \in J} x_{ej} = Y,$$

$$x_{ej} \leq C_{ej}, \quad \forall e \in E, j \in J,$$

$$\sum_{j \in J} x_{ej} U_{jt} \leq 1, \quad \forall e \in E, t \in T,$$

$$\sum_{e \in E} x_{ej} \leq S_j, \quad \forall j \in J,$$

$$\sum_{j \in J} x_{ej} \leq P_e, \quad \forall e \in E,$$

$$x_{ej} \in \{0, 1\}, \quad \forall e \in E, j \in J.$$

(3.10)

### 3.3.4 AIMMS implementation

The ID case has been implemented in AIMMS as well. The three layer flowgraph solver has been implemented as a new function. This function performs the same as the one that is used in the cases of Section 3.1 and 3.2. For this case the function is constructed such that it builds a graph of three layers with parameters connecting the different layers. In this graph an arc is added to connect the source and sink node. This arc has costs that are much bigger than any source sink path through the graph that does not uses this arc. This arc is added because there is no guarantee that all students can have their preferred number of courses, and a desired solution will maximize the number of assigned courses. Hence, the source-sink arc is only used if no more courses can be assigned. The excel read and write functionality has also been included.

The performance of the AIMMS functionality is as equivalent to the cases of of Section 3.1 and 3.2. The result is a student course assignment that maximizes the number of assigned courses while obeying the restrictions.
3.3.5 Expansion of the case

In v.d. Broek et al., 2006 additional restrictions for the ID case are proposed. However these restrictions will not be covered in this thesis. More information on these restrictions can be found in Section 7.2.

3.4 Case 4: Mourik Services

3.4.1 Description of the case

Mourik Services B.V. is a service provider for the (petro)chemical, steel, energy and environmental sectors. These technical services include consultancy, execution, maintenance and management and involve disciplines such as mechanical engineering, high-pressure and vacuum cleaning, catalyst handling, asbestos removal, environmental technology and civil engineering.

The case explained at a meeting with Mourik Services employees is in the rolling material department. At this department the planning of material and personnel is done completely manual, by means of pen and paper. The question posed was if it would be possible to have a computer aided planning tool, to make the planning more structured and efficient.

3.4.2 Problem definition

In the planning there are different types of resources (e.g. operators, cleaners, high pressure cleaning trucks and vacuum cleaning trucks). Other Mourik divisions working at client sites fax orders to the Mourik Services planner at least one day in advance. An order contains one or several order lines. Each order line can request a resource type (e.g. an operator or a cleaner) or a specific resource.

The availability of each employee is registered in an agenda. The qualifications and plant or customer specific qualifications of the personnel is registered in the database part of the Profit package, which is already in use at Mourik Services. The due dates and legalities of inspections on the material is available in Microsoft Access. The specifications and orders are available in the software package Metacom, which is already in use at Mourik Services.

More information about Mourik Services can be found at their website: http://www.mourik.com

At the moment orders arrive by fax, but it is a wish of Mourik Services to use the Metacom functionality for placing orders.
These information flows are represented in Figure 3.4.

The problem is to create a tool that enables the planner to assign material and personnel to sites in a more structured and efficient way.

3.4.3 Results

A demo version of a possible future planning application (Figure 3.5) has been demonstrated in a second meeting with Mourik Services [Mourik Services, 2007b]. It has the function to automatically couple the specific resources if they are uniquely demanded.

In the interface there are three main tables (1), (2), and (3). (1) gives the order lines that are not yet planned. (2) gives a list of specific resources that are available but have not yet been assigned to an order. (3) gives the planned orders with the requested resource and the assigned resource.

Button (A) is to assign an available resource to an unplanned order line. After this button is pressed (and the order line and resource match) the order line goes to the bottom table (3) and the resource is removed from the available resource list (2). The three bottom buttons (B), (C), and (D) are to release (un-plan) all order lines (B), release the selected order line (D) and automatically plan the non-conflicting requested specific resources (C).
Figure 3.5: Interface of the planning tool.
Chapter 4

Lagrange and flow

In Section 3.1 an MIP model was given for the simplified planning problem (3.1) and an MIP specification for the planning problem with follow up appointments (3.2).

This extended problem can be solved with an MIP solver. In this chapter an alternative is discussed. The method described here is based on solving the problem using a combination of Lagrange relaxation and flow.

This chapter first gives a general introduction to Lagrangean relaxation in Section 4.1. After that this method is used to solve a flow problem. The focus is then moved to the extended planning problem and how to solve this problem with flow. This chapter will conclude with a description of the implementation and a discussion of the results.

4.1 Introduction to Lagrange relaxation

This section will give a general introduction to Lagrangean relaxation. The information in this section is based on Section 4.3 of Bertsimas and Weismantel, 2005.

4.1.1 Matrix notation

For ease of understanding a quick introduction to matrix notation is given. Let $x_j$ be a decision variable. Then $x$ is the vector of these decision variables. The inner product of vectors $c$ and $x$ is noted with $c^T x$ which is an alternate notation for $\sum_j c_j x_j$. $A_{ij}$ is a two
dimensional parameter. With $Ax$ the matrix product between matrix $A$ and vector $x$ is meant. The result is a vector with index $i$ that for every $i$ has as value $\sum_j A_{ij}x_j$.

The constraints
\[ \sum_j A_{ij}x_j \leq b_i, \quad \forall i, \quad (4.1) \]
can be rewritten in matrix notation as
\[ Ax \leq b. \quad (4.2) \]

### 4.2 Lagrange Relaxation

Consider the following problem:

\[
\begin{align*}
\text{Min} \quad & c'x, \\
\text{s.t.} \quad & Ax \geq b, \\
& Dx \geq d, \\
& x \in \mathbb{Z}.
\end{align*}
\quad (4.3)
\]

Here $A, b, D, d, c$ have integer entries. Let $Z_{IP}$ be the optimal cost and define

\[ \mathcal{F} = \{ x \in \mathbb{Z}^n | Dx \geq d \}. \quad (4.4) \]

Next introduce a Lagrange multiplier for each of the constraints $Ax \geq b$. Let $\lambda \geq 0$ be the vector of these multipliers. For a fixed vector $\lambda$ we introduce the problem:

\[
\begin{align*}
\text{Min} \quad & c'x + \lambda'(b - Ax), \\
\text{s.t.} \quad & x \in \mathcal{F}.
\end{align*}
\quad (4.5)
\]

Denote the optimal cost by $Z(\lambda)$. If (4.3) has an optimal solution then $Z(\lambda) \leq Z_{IP}$. The function $Z(\lambda)$ is concave and piecewise linear\(^1\). Since (4.5) gives a lower bound on (4.3) it

\[^1\text{The proof of this can be found in } \text{Bertsimas and Weismantel, 2005}].
is natural to consider the tightest bound, given by the problem

\[
\begin{align*}
\text{Max} & \quad Z(\lambda), \\
\text{s.t.} & \quad \lambda \geq 0.
\end{align*}
\]  

(4.6)

Model (4.6) is referred to as the Lagrangean dual. Let \( Z_D \) be the optimal value of this problem.

Two important theorems from \[\text{Bertsimas and Weismantel, 2005}\] state that

\[
Z_D \leq Z_{IP},
\]  

(4.7)

and the optimal value \( Z_D \) is equal to the optimal cost of

\[
\begin{align*}
\text{Min} & \quad c'x, \\
\text{s.t.} & \quad Ax \geq b, \\
& \quad x \in \text{convex hull of } \mathcal{F}.
\end{align*}
\]  

(4.8)

If for a given particular value of \( \lambda \) \( Z(\lambda) \) is not \( -\infty \), then it is equal to:

\[
Z(\lambda) = \text{Min}_{k \in \mathcal{K}}(c'x^k + \lambda'(b - Ax^k)).
\]  

(4.9)

Here \( x^k, k \in \mathcal{K} \), are the extreme points of the convex hull of \( \mathcal{F} \).

### 4.2.1 Subgradient optimization algorithm

A function \( f : \mathbb{R}^n \mapsto \mathbb{R} \) is concave if and only if for any \( x^* \in \mathbb{R}^n \) there exists a vector \( s \in \mathbb{R}^n \) such that

\[
f(x) \leq f(x^*) + s'(x - x^*),
\]

for all \( x \in \mathbb{R}^n \).

Let \( f \) be a concave function. A vector \( s \) such that

\[
f(x) \leq f(x^*) + s'(x - x^*),
\]
for all $x \in \mathbb{R}^n$, is called a subgradient of $f$ at $x^*$. The set of all subgradients of $f$ at $x^*$ is denoted by $\partial f(x^*)$ and is called the subdifferential of $f$ at $x^*$.

In [Bertsimas and Weismantel, 2005] the following algorithm is given to solve an integer programming problem by means of Lagrange Relaxation.

**Algorithm 1.** The subgradient optimization algorithm.

**Input:** A nondifferentiable concave function $Z(\lambda)$.

**Output:** A maximizer $\lambda$ of $Z(\lambda)$ subject to $\lambda \geq 0$.

**Algorithm:**

1. Choose a starting point $\lambda_1^1 \geq 0$; Let $k = 1$.

2. Given $\lambda^k$, check whether $0 \in \partial Z(\lambda^k)$. If so, then $\lambda^k$ is optimal and the algorithm terminates. Otherwise, choose a subgradient $s^k \in \partial Z(\lambda^k)$ of the function $Z(\lambda^k)$.

3. Let $\lambda_{k+1} = \max \{\lambda_k + \theta s_k; 0\}$, where $\theta$ is a positive stepsize parameter. Increment $k$ and go to Step 2.

In any solution $x \in \mathcal{F}$ a valid new subgradient $s$ is the direction $\Delta \lambda = (Ax - b)$. For $\varepsilon$ small enough $Z(\lambda + \varepsilon \Delta \lambda)$ uses the same basic solution $x$ as $Z(\lambda)$. The value of $Z(\lambda + \varepsilon \Delta \lambda) = Z(\lambda) + \varepsilon (Ax - b)(b - Ax) < Z(\lambda)$ and hence it is a move in the right direction. Take care that for all $j$ it must hold that $\lambda_j \geq 0$. So, instead of $\lambda^k + \theta s^k_j$ choose max{$\lambda^k_j + \theta s^k_j$, 0} for $\lambda^k_{j+1}$. If ($s^k_j \leq 0, \forall j$) and ($\lambda_j = 0$ for $s^k_j < 0$) then $0 \in \partial Z(\lambda)$ and there is no improving direction.

### 4.3 Linking the IP model to a min cost max flow

Recall the definition of the basic planning problem (3.1) without follow up tasks:

\[
\begin{align*}
\text{Max} & \quad \sum_{e \in \mathcal{E}, j \in \mathcal{J}, t \in \mathcal{T}} x_{ejt}, \\
\text{s.t.} & \quad \sum_{e \in \mathcal{E}} x_{ejt} = F_{jt}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}, \\
& \quad \sum_{j \in \mathcal{J}} x_{ejt} \leq A_{et}, \quad \forall e \in \mathcal{E}, t \in \mathcal{T}, \\
& \quad x_{ejt} \leq C_{ej}, \quad \forall e \in \mathcal{E}, j \in \mathcal{J}, t \in \mathcal{T}, \\
& \quad x_{ejt} \in \{0, 1\}, \quad \forall e \in \mathcal{E}, j \in \mathcal{J}, t \in \mathcal{T}. 
\end{align*}
\]

(4.10)

Any feasible solution to this problem is a valid solution to the planning problem. The problem (4.10) can be interpreted as finding a min cost max flow in the graph with nodes:
source, \(v_{et}, v_{jt}, \text{sink}\),

and edges:

- \((source, v_{et})\) \(\forall e, t,\)
- \((v_{et}, v_{jt})\) \(\forall e, j, t,\)
- \((v_{jt}, \text{sink})\) \(\forall j, t,\)

with edge capacities:

- \(A_{et}\) for \((source, v_{et})\) \(\forall e, t,\)
- \(C_{ej}\) for \((v_{et}, v_{jt})\) \(\forall e, j, t,\)
- \(F_{jt}\) for \((v_{jt}, \text{sink})\) \(\forall j, t,\)

and edge costs:

- 0 for \((source, v_{et})\) \(\forall e, t,\)
- 1 for \((v_{et}, v_{jt})\) \(\forall e, j, t,\)
- 0 for \((v_{jt}, \text{sink})\) \(\forall j, t,\)

If the min cost max flow has an amount of flow equal to \(\sum_{j \in J, t \in T} F_{jt}\) then the solution of the min cost max flow is the solution to problem (4.10) with costs equal to the costs of the corresponding edges. Hence, the solution to the min cost max flow is a valid solution for our schedule. Note that a higher amount of flow than \(\sum_{j \in J, t \in T} F_{jt}\) is not possible in this graph.
4.4 The relaxed extended planning problem

Recall the IP definition of extended planning problem (3.2):

$$\begin{align*}
\text{Max} & \quad \sum_{e \in E, j \in J, t \in T} x_{ejt}, \\
\text{s.t.} & \quad \sum_{e \in E} x_{ejt} = F_{jt}, \quad \forall j \in J \setminus I^*, t \in T, \\
& \quad \sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in E, t \in T, \\
& \quad x_{ejt} \leq C_{ej}, \quad \forall e \in E, j \in J, t \in T, \\
& \quad x_{ejt} \in \{0,1\}, \quad \forall e \in E, j \in J, t \in T. 
\end{align*}$$

(4.11)

Let $\lambda_{eit_1}, \mu_{eit_1}, \nu_{eit_1}, \xi_{eit_1}$ be the Lagrange multipliers associated with the constraints defined by the last four lines of this problem definition.
Then \( Z(\lambda, \mu, \nu, \xi) \) becomes:

\[
\begin{align*}
\text{Min} & \quad \sum_{e \in E} x_{e jt}, \\
& + \sum_{e \in E, i \in I, t_1 \in T} \lambda_{e it_1} \left( \sum_{t_2 \leq t_1} x_{e it_2} - \sum_{t_2 \leq t_1+b} x_{e it_2} \right) \\
& + \sum_{e \in E, i \in I, t_1 \in T} \mu_{e it_1} \left( \sum_{t_2 \geq t_1} x_{e it_2} - \sum_{t_2 \geq t_1+1} x_{e it_2} \right) \\
& + \sum_{e \in E, i \in I, t_1 \in T} \nu_{e it_1} \left( \sum_{t_2 \leq t_1} x_{e it_2} - \sum_{t_2 \leq t_1-1} x_{e it_2} \right) \\
& + \sum_{e \in E, i \in I, t_1 \in T} \xi_{e it_1} \left( \sum_{t_2 \geq t_1} x_{e it_2} - \sum_{t_2 \geq t_1-b} x_{e it_2} \right)
\end{align*}
\]

(4.12)

s.t.

\[
\begin{align*}
\sum_{e \in E} x_{e jt} &= F_{jt}, \quad \forall j \in J \setminus I^*, t \in T, \\
\sum_{j \in J} x_{e jt} &\leq A_{et}, \quad \forall e \in E, t \in T, \\
x_{e jt} &\leq C_{e j}, \quad \forall e \in E, j \in J, t \in T, \\
x_{e jt} &\in \{0, 1\}, \quad \forall e \in E, j \in J, t \in T.
\end{align*}
\]

Problem (4.12) with fixed vectors \( \lambda, \mu, \nu, \xi \) can be solved using the min cost max flow solution as given in Section 4.3.

Let \( \bar{x} \) be the variable associated with the optimal value of this instance. Let \( Z^k \) be the solution of the \( k^{th} \) iteration. For the adjustment \( s^k \) the \( \Delta \lambda \) as given at the end of Section 4.1 can be used. This \( \Delta \lambda \) is the amount a certain constraint is not met. In this case that means per person and time interval the number of \( i \)'s that do not have an associated \( i^* \) and the number of \( i^* \)'s that do not have an associated \( i \).

Suppose an upper bound of (4.12) can be found, let \( \hat{Z} \) be this value. With the \( s^k \) as given above and a \( \theta^k \) of 1 the increase of from \( Z^{k-1} \) to \( Z^k \) when \( \lambda, \mu, \nu, \xi \) are changed with \( (s^k \theta^k) \) (given that the solution \( x \) does not change) the \( \Delta Z \), the change in \( Z(\lambda, \mu, \nu, \xi) \), can be found.

An option for \( k^{th} \) stepsize \( \theta^k \) would be \( \frac{\hat{Z} - Z^{k-1}}{2\Delta Z} \). The expected increase of \( Z \) is half the distance to the maximum value. In practice however this stepsize gives huge fluctuations. When the current solution is far from the optimal solution the stepsize is large en thus a big (not by definition better) change is made.
The stepsize used in the implementation is,

\[ \theta^k = \text{Min}\left\{ \frac{10}{k}, 1 \right\}. \]

In the beginning the solution makes huge (but controlled) changes, while after a number of iterations the stepsize becomes smaller in such a way that any solution still can be reached. The choice for this stepsize is done based on test results. This stepsize seemed the most robust in the different tested cases.

4.5 The Lagrange relaxation rewritten

For the implementation of the Lagrange relaxed flow problem (4.12) the Lagrange modifiers (\(\lambda, \mu, \nu\) and \(\xi\)) must be rewritten in terms of edge costs. As seen before variable \(x_{ejt}\) can be interpreted as the decision variable of having an edge between the node \(v_{et}\) and \(v_{jt}\).

A closer look at the cost function of problem (4.12) gives:

\[
\sum_{e \in E, i \in I, t_1 \in T} \lambda_{eit_1} \left( \sum_{t_2 \leq t_1} x_{eit_2} \right) - \sum_{e \in E, i \in I, t_1 \in T} \lambda_{eit_1} \left( \sum_{t_2 \leq t_1 + b} x_{eit_2} \right) \\
+ \sum_{e \in E, i \in I, t_1 \in T} \mu_{eit_1} \left( \sum_{t_2 \geq t_1} x_{eit_2} \right) - \sum_{e \in E, i \in I, t_1 \in T} \mu_{eit_1} \left( \sum_{t_2 \geq t_1 + 1} x_{eit_2} \right) \\
+ \sum_{e \in E, i \in I, t_1 \in T} \nu_{eit_1} \left( \sum_{t_2 \leq t_1} x_{eit_2} \right) - \sum_{e \in E, i \in I, t_1 \in T} \nu_{eit_1} \left( \sum_{t_2 \leq t_1 - 1} x_{eit_2} \right) \\
+ \sum_{e \in E, i \in I, t_1 \in T} \xi_{eit_1} \left( \sum_{t_2 \geq t_1} x_{eit_2} \right) - \sum_{e \in E, i \in I, t_1 \in T} \xi_{eit_1} \left( \sum_{t_2 \geq t_1 - b} x_{eit_2} \right). 
\]

Which can be rewritten as:

\[
\sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \geq t_2} \lambda_{eit_1} x_{eit_2} - \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \geq t_2 - b} \lambda_{eit_1} x_{eit_2} \\
+ \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \leq t_2} \mu_{eit_1} x_{eit_2} - \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \leq t_2 - 1} \mu_{eit_1} x_{eit_2} \\
+ \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \geq t_2} \nu_{eit_1} x_{eit_2} - \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \geq t_2 + 1} \nu_{eit_1} x_{eit_2} \\
+ \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \leq t_2} \xi_{eit_1} x_{eit_2} - \sum_{e \in E, i \in I, t_2 \in T} \sum_{t_1 \leq t_2 + b} \xi_{eit_1} x_{eit_2}. 
\]
And further rewriting gives:

\[
\sum_{e \in E, i \in I, t_{2} \in T} x_{eit} \left( \sum_{t_{1} \geq t_{2}} \lambda_{eit_{1}} \right) - \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \geq t_{2} - b} \lambda_{eit_{1}} \right)
+ \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2}} \mu_{eit_{1}} \right) - \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2} - 1} \mu_{eit_{1}} \right)
+ \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2}} \nu_{eit_{1}} \right) - \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2} + 1} \nu_{eit_{1}} \right)
+ \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2}} \xi_{eit_{1}} \right) - \sum_{e \in E, i \in I, t_{2} \in T} x_{eit_{2}} \left( \sum_{t_{1} \leq t_{2} + b} \xi_{eit_{1}} \right).
\]

\[ (4.15) \]

Let the variables \( \hat{\lambda}_{eit_{2}} \) and \( \hat{\lambda}_{eit_{2}^{*}} \) be defined as:

\[
\hat{\lambda}_{eit_{2}} = \sum_{t_{1} \geq t_{2}} \lambda_{eit_{1}} + \sum_{t_{1} \leq t_{2}} \mu_{eit_{1}} - \sum_{t_{1} \geq t_{2} + 1} \nu_{eit_{1}} - \sum_{t_{1} \leq t_{2} + b} \xi_{eit_{1}},
\]

\[
\hat{\lambda}_{eit_{2}^{*}} = \sum_{t_{1} \geq t_{2}} \nu_{eit_{1}} + \sum_{t_{1} \leq t_{2}} \xi_{eit_{1}} - \sum_{t_{1} \leq t_{2} - 1} \lambda_{eit_{1}} - \sum_{t_{1} \leq t_{2} - b} \mu_{eit_{1}}.
\]

Then the Lagrange problem \[4.12\] with costs in terms of edge costs is equal to:

\[
\text{Min} \quad \sum_{e \in E, j \in J, t \in T} x_{ejt} + \sum_{e \in E, i \in I, t_{2} \in T} \hat{\lambda}_{eit_{2}} x_{eit_{2}}
+ \sum_{e \in E, i^{*} \in I^{*}, t_{2} \in T} \hat{\lambda}_{eit_{2}^{*}} x_{eit_{2}^{*}}
\]

\[
\text{s.t.}
\sum_{e \in \mathcal{E}} x_{ejt} = F_{jt}, \quad \forall j \in J \setminus I^{*}, t \in T,
\sum_{j \in J} x_{ejt} \leq A_{et}, \quad \forall e \in \mathcal{E}, t \in T,
\]

\[
x_{ejt} \leq C_{ej}, \quad \forall e \in \mathcal{E}, j \in J, t \in T,
x_{ejt} \in \{0, 1\}, \quad \forall e \in \mathcal{E}, j \in J, t \in T.
\]

\[ (4.16) \]

4.6 Implementation

The implementation of the Lagrange relaxation uses a modified version of the subgradient algorithm as given in Subsection 4.2.1. In this implementation the flowsolver is used to solve the problem with fixed \( \lambda \). This flow solution is then used to derive the \( \Delta \lambda \) or change in edge costs. These two steps are repeated in the same way as in the subgradient algorithm.
4.7 Results

The Lagrange relaxation method is used to give a lower bound on the minimal value of the integer program. In this case the optimal value (if a valid solution) exists is already known, and the entire problem is based on finding this feasible solution.

For basic test cases the Lagrange relaxation with flow solver found a valid solution (a feasible schedule) within a reasonable number of iterations. When applied to the initial case of workforce planning for an administrative company (Section 3.1) the method seems to stall as can be seen in Figure 4.1.

![Figure 4.1: Graphic result of Lagrange and flow method applied to case 3.1.](image)

In Figure 4.1 three graphs are shown, the top left graph shows the fraction of employees who have a problem with matching follow up appointments at each iteration. In the top right graph the solution value of the Lagrange relaxed flow at each iteration is given. The bottom graph shows the number of follow-up constraints that are violated.

A number of different cases which resemble the initial case in complexity, size and available work over available personnel ratio have been tested and all give the same result. Lagrange stalls without giving a feasible solution. Comparing this with the greedy filling method described in Section 3.1 which gives a solution within a minute (after possibly a
few restarts), we conclude that Lagrange in this way is not a suitable solver.

As noted before, Lagrange is not meant to give a feasible solution. Although in some cases a feasible solution is found, in the practical cases Lagrange does not give a feasible solution.

4.8 The post-fix method

As seen in the previous section, the Lagrange and flow method does not provide a feasible solution to the extended administrative company case (see Subsection 3.1.5).

Typically after 1 to 3 iterations the Lagrange and flow method provides a schedule that has all the task planned with the correct amount of follow up tasks for each task. But the leading tasks and follow up tasks do not match.

In this section the possibility is considered to turn the schedule from Lagrange (that is not itself a feasible solution) into a valid schedule.

The post-fix method uses a series of relatively simple operations to turn the Lagrange output into a valid schedule. The post-fix method works in a number of steps. The input of this method is the output of the Lagrange flow method at any iteration, the not valid schedule. A schedule that is not valid is referred to as ‘table’, a leading task is referred to as $i$, a simple task is referred to as $l$. A follow up task is referred to as $i^*$. The aim of the post-fix method is to turn a ‘table’ given by Lagrange into a valid schedule. An assumption made is that the output table from Lagrange has all the tasks planned, but some of the tasks with follow up tasks and their follow up tasks are not matching. The total number of times each task is scheduled is correct. The following steps are performed in order to change the ‘table’ into a valid schedule.

1. Match as many valid $i, i^*$ pairs as possible. This can be done by simply matching a leading task $i$ with the first valid follow up task $i^*$ if such a task exists.

2. Fix the tasks $i$ and $i^*$ matched in step 1. It is not allowed to later change tasks that are already fixed.

3. Remove all the non-fixed follow up tasks $i^*$ from the ‘table’.

4. For every non fixed leading task $i$ try to find an other employee who is competent and available on the day of leading task $i$, and who is also available to do the follow up task. Assign the two tasks to him, and fix both task $i$ and $i^*$. 
5. For every non fixed leading task \( i \) try to find an other employee who has a simple task \( l \) on the day of task \( i \), and who is competent and available to do the follow up task \( i^* \). Assign task \( i \) and the follow up task \( i^* \) to him, and the reassign simple task \( l \) to the empty roster spot \( i \) left, and fix both \( i \) and it’s follow up task. This change under the condition that all competence requirements are met.

6. Repeat steps 4 and 5 till no further improvement can be found.

The output of this method is either a valid or an invalid schedule. Since it is very fast in comparison to one Lagrange iteration, after each Lagrange iteration the result of this iteration is put into the post-fix method. If this method does not provide a valid schedule another Lagrange iteration step is done.

4.9 Post-fix results and conclusions

The results are positive. The Lagrange flow with post-fix finds in our tests a valid schedule in at most 7 Lagrange iteration steps. Lagrange flow with post-fix is included in the solver comparison in Chapter 5. There the Lagrange flow with post-fix will be measured against the greedy filling and two MIP solvers.
Chapter 5

Comparison of different methods to solve scheduling problems

In the previous chapters we discussed three solution methods: greedy filling, Lagrange flow with post-fix and MIP. To compare these methods a test has been performed. In this test the greedy filling and Lagrange are compared with two MIP solvers that can be used within the AIMMS modeling environment. The two MIP solvers are XA and CPLEX. XA is the standard MIP solver of AIMMS. CPLEX is an industrial MIP solver that is generally seen as the top of the bill MIP solver. For more information on XA see: http://www.sunsetsoft.com/, for CPLEX see: http://www.ilog.com/products/cplex/, and for more information about AIMMS licence costs see: http://www.aimms.com/aimms/download/pricing/aimms3.pricing_eur.jan2007.pdf

Besides the comparison of the greedy and Lagrange approach with the MIP solvers another aim of this test was to see how well the default MIP solver of AIMMS (XA) performed compared to CPLEX when solving MIP’s of scheduling problems. Another point of interest is to see if all solvers achieve the same results. For test purposes the solvers are bound in time. A maximum runtime of 20 minutes per method is chosen. The possible outcomes of a method are:

Success: The method found a feasible solution to the schedule.  
Fail: The method has stopped and did not find a feasible solution.

5.1 Test cases

For the test a number of test cases have been constructed. All the test cases use the formulation of the administrative company (Section 3.1). A case generator is created to
generate a case based on 5 parameters:

- number of employees,
- number of tasks,
- number of timeslots,
- number of leading tasks. This number must be less than the half the total number of tasks rounded down, since follow up tasks are seen as different tasks from their leading task,
- rate of utilization. The ratio of work to be carried out divided by the number of available employees.

A case is then characterized by these five numbers.

5.2 Result

The test consists of eight different settings. For each of these settings ten feasible cases are generated. For two of the eight settings also ten infeasible cases are generated, to see how solvers perform on infeasible problems.

In Tables 5.1, 5.2, 5.3, and 5.4 (at the end of this chapter) the results of the test are presented. The times are all measured in seconds and include the time it took to render the Gantt chart object as well as any time it took to convert the data to the right format. Since both greedy filling and Lagrange include random ordering of the resources these solvers are run twice to see the effect of randomization. Behind each result there is a letter denoting the status of the result. An S for Succes and an F for Fail.
5.3 Conclusion

In all the cases the CPLEX solver performs better than the XA solver. In most cases the time it took XA is acceptable in solving the cases. When the number of follow up tasks increases the time it takes XA can increase to over 20 minutes, the used time limit. In all these cases CPLEX finished within a minute.

The Lagrange and flow solver is generally performing slower than the greedy fill solver although in some very special cases the Lagrange flow halts earlier because the flowgraph is infeasible and hence the schedule problem has no feasible solution. When given an infeasible case the greedy fill solver will halt when the loopmaximum is reached. Greedy fill is not able to detect infeasible cases.

For the cases in which the utilization rate is increased to 0.85 all solvers finish within seconds. The greedy solver is not suited for this situation since it rarely produces a feasible schedule. When the utilization rate is increased to 0.9 the Lagrange solver also fails to find a feasible schedule.

Overall it seems that a combination of different solvers is recommended since for different cases a different method performs best. The CPLEX solver performs very well in all cases, but a combination of greedy fill, Lagrange and XA will usually find the solution (if one exists).

A toolkit for solving scheduling problems without the use of CPLEX is possible. Some specific cases however may require CPLEX since there exist cases that only got a feasible schedule from CPLEX. A trade-off has to be made between time and reliability on one side and the larger licence cost of CPLEX on the other side when considering to include CPLEX in the solver options.
<table>
<thead>
<tr>
<th>Case</th>
<th>Greedy I</th>
<th>Greedy II</th>
<th>Lagrange I</th>
<th>Lagrange II</th>
<th>XA</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 employees</td>
<td>1</td>
<td>38 F</td>
<td>41 F</td>
<td>21 S</td>
<td>22 S</td>
<td>19 S</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>30 S</td>
<td>7 S</td>
<td>21 S</td>
<td>20 S</td>
<td>6 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>5 S</td>
<td>5 S</td>
<td>20 S</td>
<td>21 S</td>
<td>13 S</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>5 S</td>
<td>5 S</td>
<td>17 S</td>
<td>17 S</td>
<td>4 S</td>
</tr>
<tr>
<td>0.8 utilization</td>
<td>5</td>
<td>5 S</td>
<td>4 S</td>
<td>21 S</td>
<td>22 S</td>
<td>4 S</td>
</tr>
<tr>
<td>feasible</td>
<td>6</td>
<td>11 S</td>
<td>9 S</td>
<td>19 S</td>
<td>20 S</td>
<td>7 S</td>
</tr>
<tr>
<td>7</td>
<td>4 S</td>
<td>5 S</td>
<td>19 S</td>
<td>20 S</td>
<td>5 S</td>
<td>3 S</td>
</tr>
<tr>
<td>8</td>
<td>5 S</td>
<td>8 S</td>
<td>82 S</td>
<td>83 S</td>
<td>5 S</td>
<td>2 S</td>
</tr>
<tr>
<td>9</td>
<td>5 S</td>
<td>5 S</td>
<td>21 S</td>
<td>20 S</td>
<td>10 S</td>
<td>2 S</td>
</tr>
<tr>
<td>10</td>
<td>15 S</td>
<td>11 S</td>
<td>22 S</td>
<td>22 S</td>
<td>9 S</td>
<td>3 S</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>95 F</td>
<td>65 S</td>
<td>39 S</td>
<td>98 S</td>
<td>1209 F</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>22 S</td>
<td>18 S</td>
<td>39 S</td>
<td>40 S</td>
<td>1209 F</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>10 S</td>
<td>52 S</td>
<td>410 S</td>
<td>446 S</td>
<td>237 S</td>
</tr>
<tr>
<td>3 leading tasks</td>
<td>4</td>
<td>15 S</td>
<td>45 S</td>
<td>40 S</td>
<td>38 S</td>
<td>1209 F</td>
</tr>
<tr>
<td>0.8 utilization</td>
<td>5</td>
<td>9 S</td>
<td>10 S</td>
<td>36 S</td>
<td>78 S</td>
<td>414 S</td>
</tr>
<tr>
<td>feasible</td>
<td>6</td>
<td>15 S</td>
<td>65 S</td>
<td>39 S</td>
<td>39 S</td>
<td>1208 F</td>
</tr>
<tr>
<td>7</td>
<td>154 S</td>
<td>467 F</td>
<td>849 F</td>
<td>903 F</td>
<td>270 S</td>
<td>5 S</td>
</tr>
<tr>
<td>8</td>
<td>54 S</td>
<td>59 S</td>
<td>84 S</td>
<td>37 S</td>
<td>1209 F</td>
<td>11 S</td>
</tr>
<tr>
<td>9</td>
<td>37 S</td>
<td>41 S</td>
<td>140 S</td>
<td>140 S</td>
<td>1209 F</td>
<td>13 S</td>
</tr>
<tr>
<td>10</td>
<td>27 S</td>
<td>35 S</td>
<td>78 S</td>
<td>79 S</td>
<td>362 S</td>
<td>7 S</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>16 S</td>
<td>38 S</td>
<td>140 S</td>
<td>150 S</td>
<td>1213 F</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>128 S</td>
<td>249 F</td>
<td>1702 F</td>
<td>1706 F</td>
<td>1213 F</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>15 S</td>
<td>38 S</td>
<td>297 S</td>
<td>56 S</td>
<td>1212 F</td>
</tr>
<tr>
<td>4 leading tasks</td>
<td>4</td>
<td>369 F</td>
<td>314 F</td>
<td>1737 F</td>
<td>1753 F</td>
<td>1214 F</td>
</tr>
<tr>
<td>0.8 utilization</td>
<td>5</td>
<td>1208 F</td>
<td>1017 F</td>
<td>1731 F</td>
<td>1727 F</td>
<td>1213 F</td>
</tr>
<tr>
<td>feasible</td>
<td>6</td>
<td>983 F</td>
<td>809 F</td>
<td>1658 F</td>
<td>1645 F</td>
<td>1212 F</td>
</tr>
<tr>
<td>7</td>
<td>16 S</td>
<td>87 S</td>
<td>58 S</td>
<td>57 S</td>
<td>1213 F</td>
<td>59 S</td>
</tr>
<tr>
<td>8</td>
<td>434 S</td>
<td>498 S</td>
<td>146 S</td>
<td>56 S</td>
<td>1212 F</td>
<td>26 S</td>
</tr>
<tr>
<td>9</td>
<td>2260 F</td>
<td>2261 F</td>
<td>1635 F</td>
<td>1658 F</td>
<td>1212 F</td>
<td>28 S</td>
</tr>
<tr>
<td>10</td>
<td>49 S</td>
<td>321 S</td>
<td>1688 F</td>
<td>152 S</td>
<td>1213 F</td>
<td>25 S</td>
</tr>
</tbody>
</table>

Table 5.1: Results of the test in seconds.
<table>
<thead>
<tr>
<th>Case</th>
<th>Greedy I</th>
<th>Greedy II</th>
<th>Lagrange I</th>
<th>Lagrange II</th>
<th>XA</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 employees</td>
<td>1</td>
<td>13 S</td>
<td>13 S</td>
<td>54 S</td>
<td>55 S</td>
<td>9 S</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>13 S</td>
<td>14 S</td>
<td>61 S</td>
<td>60 S</td>
<td>12 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>14 S</td>
<td>14 S</td>
<td>59 S</td>
<td>59 S</td>
<td>13 S</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>14 S</td>
<td>13 S</td>
<td>53 S</td>
<td>53 S</td>
<td>11 S</td>
</tr>
<tr>
<td>0.5 utilization</td>
<td>5</td>
<td>14 S</td>
<td>14 S</td>
<td>62 S</td>
<td>61 S</td>
<td>13 S</td>
</tr>
<tr>
<td>feasible</td>
<td>6</td>
<td>13 S</td>
<td>13 S</td>
<td>56 S</td>
<td>55 S</td>
<td>9 S</td>
</tr>
<tr>
<td>7</td>
<td>14 S</td>
<td>14 S</td>
<td>63 S</td>
<td>63 S</td>
<td>13 S</td>
<td>7 S</td>
</tr>
<tr>
<td>8</td>
<td>13 S</td>
<td>14 S</td>
<td>57 S</td>
<td>56 S</td>
<td>13 S</td>
<td>7 S</td>
</tr>
<tr>
<td>9</td>
<td>14 S</td>
<td>13 S</td>
<td>57 S</td>
<td>56 S</td>
<td>11 S</td>
<td>7 S</td>
</tr>
<tr>
<td>10</td>
<td>14 S</td>
<td>14 S</td>
<td>55 S</td>
<td>56 S</td>
<td>12 S</td>
<td>8 S</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>16 S</td>
<td>15 S</td>
<td>62 S</td>
<td>62 S</td>
<td>11 S</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>17 S</td>
<td>16 S</td>
<td>69 S</td>
<td>69 S</td>
<td>12 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>16 S</td>
<td>17 S</td>
<td>66 S</td>
<td>67 S</td>
<td>13 S</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>16 S</td>
<td>16 S</td>
<td>64 S</td>
<td>64 S</td>
<td>13 S</td>
</tr>
<tr>
<td>feasible</td>
<td>5</td>
<td>16 S</td>
<td>16 S</td>
<td>67 S</td>
<td>67 S</td>
<td>13 S</td>
</tr>
<tr>
<td>7</td>
<td>16 S</td>
<td>16 S</td>
<td>70 S</td>
<td>69 S</td>
<td>13 S</td>
<td>9 S</td>
</tr>
<tr>
<td>8</td>
<td>16 S</td>
<td>15 S</td>
<td>59 S</td>
<td>58 S</td>
<td>12 S</td>
<td>9 S</td>
</tr>
<tr>
<td>9</td>
<td>16 S</td>
<td>15 S</td>
<td>64 S</td>
<td>63 S</td>
<td>13 S</td>
<td>9 S</td>
</tr>
<tr>
<td>10</td>
<td>16 S</td>
<td>16 S</td>
<td>68 S</td>
<td>68 S</td>
<td>13 S</td>
<td>9 S</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>25 S</td>
<td>24 S</td>
<td>93 S</td>
<td>92 S</td>
<td>1218 F</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>24 S</td>
<td>24 S</td>
<td>82 S</td>
<td>82 S</td>
<td>206 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>24 S</td>
<td>24 S</td>
<td>87 S</td>
<td>87 S</td>
<td>773 S</td>
</tr>
<tr>
<td>3 leading tasks</td>
<td>4</td>
<td>24 S</td>
<td>25 S</td>
<td>90 S</td>
<td>94 S</td>
<td>300 S</td>
</tr>
<tr>
<td>0.5 utilization</td>
<td>5</td>
<td>24 S</td>
<td>25 S</td>
<td>89 S</td>
<td>90 S</td>
<td>340 S</td>
</tr>
<tr>
<td>feasible</td>
<td>6</td>
<td>22 S</td>
<td>22 S</td>
<td>72 S</td>
<td>157 S</td>
<td>52 S</td>
</tr>
<tr>
<td>7</td>
<td>24 S</td>
<td>24 S</td>
<td>89 S</td>
<td>89 S</td>
<td>391 S</td>
<td>11 S</td>
</tr>
<tr>
<td>8</td>
<td>24 S</td>
<td>25 S</td>
<td>89 S</td>
<td>89 S</td>
<td>750 S</td>
<td>11 S</td>
</tr>
<tr>
<td>9</td>
<td>24 S</td>
<td>23 S</td>
<td>85 S</td>
<td>85 S</td>
<td>1217 F</td>
<td>16 S</td>
</tr>
<tr>
<td>10</td>
<td>24 S</td>
<td>33 S</td>
<td>77 S</td>
<td>76 S</td>
<td>63 S</td>
<td>11 S</td>
</tr>
</tbody>
</table>

Table 5.2: Results of the test in seconds.
<table>
<thead>
<tr>
<th>Case</th>
<th>Greedy I</th>
<th>Greedy II</th>
<th>Lagrange I</th>
<th>Lagrange II</th>
<th>XA</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 employees</td>
<td>1</td>
<td>5 F</td>
<td>5 F</td>
<td>4 S</td>
<td>4 S</td>
<td>11 S</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>5 F</td>
<td>5 F</td>
<td>4 S</td>
<td>5 S</td>
<td>23 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>1 S</td>
<td>6 F</td>
<td>4 S</td>
<td>4 S</td>
<td>13 S</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>1 S</td>
<td>1 S</td>
<td>5 S</td>
<td>4 S</td>
<td>6 S</td>
</tr>
<tr>
<td>0.85 utilization</td>
<td>5</td>
<td>6 S</td>
<td>5 F</td>
<td>4 S</td>
<td>4 S</td>
<td>21 S</td>
</tr>
<tr>
<td>feasible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4 F</td>
<td>4 F</td>
<td>5 S</td>
<td>5 S</td>
<td>9 S</td>
<td>0 S</td>
</tr>
<tr>
<td>7</td>
<td>7 F</td>
<td>5 F</td>
<td>5 S</td>
<td>4 S</td>
<td>9 S</td>
<td>1 S</td>
</tr>
<tr>
<td>8</td>
<td>5 S</td>
<td>5 S</td>
<td>5 S</td>
<td>5 S</td>
<td>16 S</td>
<td>1 S</td>
</tr>
<tr>
<td>9</td>
<td>5 F</td>
<td>5 F</td>
<td>5 S</td>
<td>5 S</td>
<td>20 S</td>
<td>1 S</td>
</tr>
<tr>
<td>10</td>
<td>3 F</td>
<td>3 F</td>
<td>5 S</td>
<td>5 S</td>
<td>9 S</td>
<td>1 S</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>5 F</td>
<td>15 S</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>4 F</td>
<td>3 F</td>
<td>4 F</td>
<td>5 F</td>
<td>8 S</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>4 F</td>
<td>22 S</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>10 S</td>
</tr>
<tr>
<td>0.9 utilization</td>
<td>5</td>
<td>4 F</td>
<td>3 F</td>
<td>5 F</td>
<td>4 F</td>
<td>12 S</td>
</tr>
<tr>
<td>feasible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>6 S</td>
<td>0 S</td>
</tr>
<tr>
<td>7</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>11 S</td>
<td>1 S</td>
</tr>
<tr>
<td>8</td>
<td>4 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>19 S</td>
<td>1 S</td>
</tr>
<tr>
<td>9</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>40 S</td>
<td>1 S</td>
</tr>
<tr>
<td>10</td>
<td>2 F</td>
<td>1 F</td>
<td>5 F</td>
<td>3 F</td>
<td>3 S</td>
<td>0 S</td>
</tr>
</tbody>
</table>

Table 5.3: Results of the test in seconds.
Table 5.4: Results of the test in seconds.

<table>
<thead>
<tr>
<th>Case</th>
<th>Greedy I</th>
<th>Greedy II</th>
<th>Lagrange I</th>
<th>Lagrange II</th>
<th>XA</th>
<th>CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 employees</td>
<td>1</td>
<td>5 F</td>
<td>4 F</td>
<td>4 F</td>
<td>4 F</td>
<td>6 F</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>1 F</td>
<td>2 F</td>
<td>4 F</td>
<td>3 F</td>
<td>2 F</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>2 F</td>
<td>1 F</td>
<td>4 F</td>
<td>3 F</td>
<td>2 F</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>3 F</td>
<td>7 F</td>
</tr>
<tr>
<td>0.85 utilization</td>
<td>5</td>
<td>1 F</td>
<td>1 F</td>
<td>4 F</td>
<td>3 F</td>
<td>1 F</td>
</tr>
<tr>
<td>infeasible</td>
<td>6</td>
<td>3 F</td>
<td>2 F</td>
<td>4 F</td>
<td>4 F</td>
<td>4 F</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>2 F</td>
<td>1 F</td>
<td>4 F</td>
<td>3 F</td>
<td>2 F</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1 F</td>
<td>2 F</td>
<td>4 F</td>
<td>4 F</td>
<td>2 F</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>1 F</td>
<td>2 F</td>
<td>3 F</td>
<td>4 F</td>
<td>2 F</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2 F</td>
<td>2 F</td>
<td>4 F</td>
<td>3 F</td>
<td>2 F</td>
</tr>
<tr>
<td>30 employees</td>
<td>1</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>5 F</td>
<td>15 F</td>
</tr>
<tr>
<td>10 task types</td>
<td>2</td>
<td>4 F</td>
<td>3 F</td>
<td>4 F</td>
<td>5 F</td>
<td>8 F</td>
</tr>
<tr>
<td>30 timeslots</td>
<td>3</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>4 F</td>
<td>22 F</td>
</tr>
<tr>
<td>1 leading task</td>
<td>4</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>10 F</td>
</tr>
<tr>
<td>0.9 utilization</td>
<td>5</td>
<td>4 F</td>
<td>3 F</td>
<td>5 F</td>
<td>4 F</td>
<td>12 F</td>
</tr>
<tr>
<td>infeasible</td>
<td>6</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>6 F</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>11 F</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>19 F</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>3 F</td>
<td>3 F</td>
<td>4 F</td>
<td>4 F</td>
<td>40 F</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2 F</td>
<td>1 F</td>
<td>5 F</td>
<td>3 F</td>
<td>3 F</td>
</tr>
</tbody>
</table>
Chapter 6

Framework of the general toolkit

In Figure 6.1 a schematic overview is given of the different modules of the general toolkit. There are end users, the resources of the scheduling problem, and there is a planner who works on the system from different angles.

The application itself has three modules each designed for a different task. The end users communicate via a web interface. This web interface communicates with a database in which all the information is stored. The planner operates AIMMS directly, and from AIMMS he is able to import and export data to and from the database. He is also able to modify the input and make changes to the output schedule within the AIMMS application.

Once AIMMS has produced a schedule that the planner agrees upon, he can export it to the database (or export it from AIMMS to an Excel file). After that end users are able to retrieve the schedule from the database. The end users are not able to change the schedule in the database. Using a security module it can be controlled which data can be retrieved by each user.
6.1 Interfaces

The system has four interfaces. Two of these interfaces are Graphical User Interfaces (GUI’s) and two of these interfaces are between two modules of the application. The list below describes the options of each interface.

Interface A

*The end user is allowed to:*
- Read the schedule from the web application.
- Read his previously entered availability information.
- Write his availability into the database.

*In the security module it can be defined for each user what data can be accessed.*

Interface B

*The web module can:*
- Read the schedule.
- Read resource availability.
- Write availability of end users.

Interface C

*The AIMMS module can:*
- Read the complete database.
- Write the complete database.

Interface D

*The planner is allowed to:*
- Let the AIMMS module read the database.
- Modify the availability data in AIMMS.
- Modify the competence data in AIMMS.
- Generate a planning in AIMMS.
- Modify the planning in AIMMS.
- Save the data from AIMMS in the database.

### 6.2 Modules

The web interface module is built in Java using Java Server Pages (JSP) to directly get and put the information into the database. A web interface has been created that supports both the administrative company and the printing company case, the cases from Section 3.1 and Section 3.2 respectively. The web interface is password protected. A user can only access and modify his own data (availability, scheduled tasks) unless otherwise specified in the security module of the web interface.

Each of those two cases has a separate Microsoft Access database with its own Open Database Connectivity (ODBC) link. ODBC is an interface to databases that is aimed to make it independent of programming languages, database systems, and operating systems. Each case has its own logical table and column names. It is also possible to put all data of different cases/companies in one database while maintaining the access restrictions per user to distinguish between companies. The differences between one database for all the cases and a separate database for each case are closely related to the choice of web server business model that is discussed in Subsection 6.2.5.

The core planning application is built in AIMMS. This allows for easy and flexible interfaces (like Gantt charts) into the planning tool while at the same time offering a solid environment for optimization algorithms including linear and integer programming. The planner uses the AIMMS GUI, which provides flexibility and power to generate and adapt a schedule while maintaining the integrity of the solution methods. AIMMS provides functionality to allow password protected access to the full code or only to the interfaces.

#### 6.2.1 AIMMS

The AIMMS module is the essence of the application. In this module all the input is gathered and the schedule is made. In order to make the application as generic as possible, the AIMMS project has been divided into libraries. Libraries are a new feature of AIMMS 3.7. They are intended to create a multi developer environment for AIMMS and to easily exchange cores of AIMMS code between different projects. The main idea of this environment is the ability to divide your AIMMS project into different libraries, where each library
has a specific task.

For each of the libraries you can define which part is accessible by other AIMMS libraries. Each library is saved into its own set of project files. The latter allows the creation of standard libraries that perform common functions that can be quickly used in new projects providing common tasks.

For the generic toolkit a number of such libraries are created:

**Flow Optimization** A library for solving a schedule problem by converting it to a flow problem and subsequently solving this flow problem.

**Gantt chart** A library for creating Gantt chart objects.

**Case** A library that contains only the specific information for the case.

![Figure 6.2: A schematic overview of the AIMMS libraries and communication.](image)

For every task it has to be considered how generic it can be made, and how much time and effort is saved by using a separate library. For instance the database read and write functions need to be tailor made to fit the database of the project. Hence, no time or effort can be saved by making it general, since the information you need to enter in the database functions, you also need to give it to the library. The same argument goes for the read/write function from Excel. Furthermore, note that once the data model is defined the implementation of the AIMMS module is straightforward and fast.

In Table 6.1 a list of all functions is given. In the Appendix a detailed description of these functions and their required parameters is given.
<table>
<thead>
<tr>
<th>Library</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>TwoLayerFlow</td>
<td>Solves a flow problem consisting of two layers.</td>
</tr>
<tr>
<td></td>
<td>ThreeLayerFlow</td>
<td>Solves a flow problem consisting of three layers.</td>
</tr>
<tr>
<td></td>
<td>LagrangeTwoLayerFlow</td>
<td>Solves a Lagrange relaxed flow problem consisting of two layers.</td>
</tr>
<tr>
<td></td>
<td>DllSolveMinCostFlow</td>
<td>Function needed in AIMMS to execute the Goldberg flow library.</td>
</tr>
<tr>
<td></td>
<td>Bottleneck</td>
<td>Finds the bottleneck when a feasible schedule cannot be found.</td>
</tr>
<tr>
<td></td>
<td>LagrangePostFix</td>
<td>The post-fix steps used to speed up the LagrangeTwoLayerFlow function.</td>
</tr>
<tr>
<td>Gantt</td>
<td>EraseGanttChart</td>
<td>Empties the data of the Gantt chart object.</td>
</tr>
<tr>
<td></td>
<td>DisplayGanttChart</td>
<td>Fills the data of the Gantt chart object.</td>
</tr>
<tr>
<td>Case files</td>
<td>ReadDatabase</td>
<td>Read the case data from the database.</td>
</tr>
<tr>
<td></td>
<td>WriteDatabase</td>
<td>Write the schedule to the database.</td>
</tr>
<tr>
<td></td>
<td>ComputeSchedule</td>
<td>Compute the schedule by means of flow.</td>
</tr>
<tr>
<td></td>
<td>ExcelRead</td>
<td>Read the case data from Excel.</td>
</tr>
<tr>
<td></td>
<td>ExcelWrite</td>
<td>Write the schedule to Excel.</td>
</tr>
<tr>
<td></td>
<td>SolveMIP_CPLEX</td>
<td>Compute the schedule by means of the CPLEX MIP solver.</td>
</tr>
<tr>
<td></td>
<td>SolveMIP_XA</td>
<td>Compute the schedule by means of the XA MIP solver.</td>
</tr>
<tr>
<td>Administrative</td>
<td>SolveGreedy</td>
<td>Compute the schedule by means of the greedy fill method.</td>
</tr>
<tr>
<td></td>
<td>SolveLagrange</td>
<td>Compute the schedule by means of the Lagrange relaxed flow method with post fix.</td>
</tr>
</tbody>
</table>

Table 6.1: Overview of the different functions created.

### 6.2.2 Database

In this project a Microsoft Access database is used. This mainly because of the support by both Java and AIMMS as well as the availability and knowledge at CQM.

For a single case this database is sufficient because there is little overhead and it can be easily installed on a Microsoft Windows desktop machine. If the business model, which is discussed in Section 6.2.5, would require to have all the cases saved and run on one dedicated machine alternatives have to be considered. Microsoft Access databases do not perform very well with large data sets and have a hard data limit of 2 GB.
A MySQL server on a UNIX based system would be more suitable for the larger task at hand. Java has support for SQL databases as well. AIMMS itself can access the database over ODBC. AIMMS has the restriction that it can only display the GUI in a Windows environment. A link between the windows system and a UNIX based system has to be made in order to run a dedicated database machine, which is outside the scope of this thesis.

6.2.3 JSP web interface

In this part the different employees can enter and change their availability. Also they can check their schedule for the coming period.

The web interface consists of a few libraries. In the web interfaces we refer to libraries as pages. This because each library typically is programmed on a different output web page. All these pages communicate with a single java class that handles the database connection and queries.

6.2.4 Different JSP pages

![Diagram of JSP pages and their interaction](image.png)

Figure 6.3: The different JSP pages and their interaction.
For the web interface a database connection class is created. This class supports database extraction and insertions through a simple set of functions. These functions are:

**ReadNames:** This function returns a vector with the available names.

**ReadTimeslots:** This function returns a vector with the available timeslots.

**ReadAvailability:** This function returns the availability for all timeslots for a selected employee.

**ReadSchedule:** This function returns the schedule as it is stored in the database.

**UpdateDataBase:** This function changes the availability data in the database.

A set of web pages is created to call these database class functions and display it to the end user. The webpages that are created are given in Figure 6.3.

### 6.2.5 Web server business models

The web interface as GUI for the database gives rise to two very different business models. The choice for any one of these models has different implications for the entire system. In this subsection a comparison between the two business models is given. For each of the two models a description is given as well as the advantages and the disadvantages.

The two models are:

**Decentral servers**
This model puts each project on its own server. Most likely this server is the computer on which the planner is doing his job. This means that all the employees can access the web interface from anywhere on the web and on top of this the planner can do all his work directly on the server in a planning application.

**Central server**
This model puts all the projects on one server. This server is most likely hosted by professional webhosting. This means that the employees have to interface with the planning application by means of the web interface. For the planner there is the choice of either using a web interface or a remote desktop login.
Decentral servers

An advantage of decentral servers is that each company has its own server. As a result the planner has full access to the planning software. This means that a planning GUI can be constructed in AIMMS. The AIMMS software package has a lot of build in functions and display options to quickly and efficiently produce a clear and user friendly interface. The computational capacity of the server is only used by one planning application which is always available.

A disadvantage of decentral servers is that each company needs to host their own website. Also they need to have their own AIMMS licence including licences for MIP solvers. This licence is for a piece of software that is used only when the planner is creating a new schedule.

Another result of the decentral servers approach is that each server has its own database. This database can be tailor made for the specific planning problem of this company. And advantage of this is that each database can be optimized for performance, but a disadvantage is that for each server a database must be created and maintained.

Central server

A central server results in the central maintenance of only one system. A result of a central system is that all the planning problem holders have to share the computational power of one server. This can result in waiting times before the planning can be generated. Since there is only one server running the planning software there are several options for the planning software. Choices can be made in how many planning applications can compute a schedule at the same time and what solvers to include in the planning software. The licences can be shared by all the problem holders. This can result in an increase of solver options while keeping the costs low.

The planner has to use a web based interface in order to communicate with the planning software, and some of the interface powers of AIMMS are lost due to this. The planner can also use a remote desktop connection to the server in order to keep the AIMMS GUI.

In Table 6.2 a comparison is given between the two server models. Depending on the needs and preferences one of the models, or a mixture, can be chosen. The choice of the web server model greatly influences the resulting planning program both in GUI and performance.
<table>
<thead>
<tr>
<th></th>
<th>Decentral</th>
<th>Central</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solver costs</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Database speed</td>
<td>++</td>
<td>−</td>
</tr>
<tr>
<td>Full access to server</td>
<td>++</td>
<td>−</td>
</tr>
<tr>
<td>Web interface</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>Planner interface</td>
<td>++</td>
<td>+/ + +</td>
</tr>
</tbody>
</table>

Table 6.2: A comparison between the two server models.
Chapter 7

Results and Conclusions

7.1 Primary results

In the introduction (see Chapter [1]) three categories of questions were proposed for this master’s thesis. Based on the report an answer is formulated to each of the questions posed in the introduction.

7.1.1 Algorithmic questions

A lot of problems in their basic form can be solved by means of a flow algorithm. Case specific requirements are very likely to complicate a problem in such a way that it is no longer possible to directly use a flow model solver.

There are possibilities to extend the flow solver to greedy filling with a flow follow-up or flow in combination with Lagrange relaxation such that the problem including these requirements is still solveable with flow.

The scheduling problems reviewed in this thesis can all be solved by using MIP. To solve MIP models several solvers are available. The performance varies between the methods, as do the costs of licence fees for these solvers.

In conclusion, it is possible to build a toolkit to solve planning problems by means of flow or with algorithmic methods based on flow. This toolkit can be build versatile enough to also include other methods for solving e.g. MIP solvers.
7.1.2 Technical questions

The creation of standard components in AIMMS offers prospects for a general planning toolkit. Computations and visualizations can be generalized to such a level that they can be implemented independent of a case and used as component in future projects.

Several functions like reading and writing the case information from both Excel and a database require that many case specific settings that the standard AIMMS functionality is sufficient for this task. The parameters and input needed for these functions is the same as would be needed by the generalized read-write module and therefore case specific read-write functions are proposed.

The interface for the employees planned by the planning tool is web-based. A web-based GUI enables the end users to log in from any location on the web and review their assigned tasks as well as change their availability for the coming periods.

7.1.3 Market questions

A software package that offers the same functionality as described in this thesis is not present in the market. Either the automatic planning is present in a package that offers complete Enterprise Resource Planning (ERP) solutions, the package only offers computer aided scheduling (opposed to fully automatic scheduling), or the package is highly market segment or branch specific.

The cases used in this thesis are all real life planning problems encountered either by CQM or Eindhoven University of Technology. A market seems to exist for tailor made planning solutions that come at a reasonable price. AIMMS offers the functionality to create custom made planning solutions that use generic functions for algorithms and visualization.

7.2 Recommendations for further research

The ID case of Section 3.3

In Section 3.2 of [v.d. Broek et al., 2006] a number of additional properties of the problem are covered. The toolkit and AIMMS functionality can also be extended to include the following features:

- Lower bounds on the number of students that can participate in a course.
• Plan more than one period of classes at the same time.

• Courses with higher workload. This results in a course having weight two in the constraint of maximum number of courses a student can take.

• Courses with two meetings per week. A course then needs two timeslots instead of one.

• Urgent cases, where a student has to get a certain course assigned.

Any one of these extensions makes the problem infeasible with flow algorithms. Different solutions like Lagrange and flow need to be constructed.

The web interface

Additional work can be done on the web interface. Currently the password is matched against a word that is hard coded in the code, and the same word is used for each account. A one way function can be used to match passwords against the result stored in the database.

The database is currently a Microsoft Access database that only runs on Windows based systems. In order to run the system on a dedicated UNIX system other database systems have to be implemented.

Solvers and methods

Research can be done to include more solvers and methods. There is a wide variety of solvers available for AIMMS and only two of these solvers have been considered in this thesis.

The AIMMS front-end

The case specific part of the toolkit can be made into a customer specific GUI that includes buttons for all the functions implemented as well as links to the Gantt chart object and other graphical representations.
Bibliography


Appendix A

Internet search results

Adecco E-tools

http://www.adecco.nl/etools.asp
Software is based on Rostar CAS (by Paralax)

Arpa Interbouw

http://www.plazoo.com/nl/topnews
http://www.arpa-intrabouw.nl
Software is very specific for the building business. Not generic enough to consider in this search.
Bit by bit information systems

![Schematic overview of the IRIS-software.](http://www.bitbybit-is.nl)

**Website:** [http://www.bitbybit-is.nl](http://www.bitbybit-is.nl)

**Size:**

**Customers:** TU-Delft, SGS, TU/e, LUMC, Haagse Hogeschool, Dutch Royal Navy, Diagnostics 4 Health, Erasmus MC, UMC St Radboud

**Branches:**

**Software:** IRIS-Planning

**Key focus:** Yes

**Extra functions:** Reservations

**Cost:** Yearly service contract - they offer a highly appealing service contract with real-time troubleshooting and aid. Their planners are also available for hire at favorable rates!

**Web Function:** Yes, users can view and manipulate their own timetable with Iris Web-View.

**Demo Available:** No
Invision software

[Figure A.2: Screenshot of the AutoScheduler software.]


Size:

Customers: ABN-Amro, BMW Financial Services, Postbank, Union Investment, Wesbank, TNT GmbH, Asfinag, Frapport Cargo Services, Deutsche Telekom AG, T-Mobile Austria, Telefonia Dialog, IKEA Deutschland Otto nederland, Neckermann Versand, Adidas

Branches: Finance, Transport & Logistics, Telecommunications, Retail, Utilities, Manufacturing, Public services, Service providers, Tourism & leisure, Insurances & Health Funds, Media

Software: AutoScheduler

Key focus: Yes

Extra functions: Unlimited options when it comes to multi-site planning.

Cost:

Web Function: No

Demo Available: No

Microsoft Navision Axapta

[No picture available]


Price Range:$100,000 - $300,000. Hence the software is not within the scope of this search. However a demo is available for download.
Mijnrooster Open Source rooster planning

Figure A.3: Screenshot of the MijnRooster software.

http://www.mijnrooster.nl/Functionaliteit/Techniek.html
No automatic planning options. Only focus on manual planning.

OM Partners

Figure A.4: Graphic representation of the OM Partners software.
Website:  http://www.ompartners.com

Customers: Akzo Nobel, Aventis, BASF, British American Tabacco, Campina, Corus, Danone, ExxonMobil, Friesland Foods, Johnson & Johnson, Michelin, Leaf Holland, L’Oreal, Otor, Samsonite, Sara Lee, Yves Saint Laurent

Branches: Flow Shop and Semi Process

Software: OMP Supply Chain Suite

Key focus: No

Extra functions: Designer, Schedules, Forecaster, Order Manager

Cost:

Web Function: Yes, OMP Portal enables internet-based collaborative applications and reporting.

Demo Available: No

Ortec (Harmony)

Figure A.5: Screenshot of the Harmony software package.
Website: [http://nl.ortec.com](http://nl.ortec.com)
Size: 500 employees
Customers: Albert.nl, TNT, Muller Transport, Pontmeyer, Schenker
Branches: Facility services, Healthcare, Construction & Industry, Air Traffic, Pension Funds, Public Transportation, Logistics, Insurance, Housing Corporations
Software: Harmony
Key focus: Yes (key focus of Harmony)
Extra functions: Time registration, web access, stageplanning, facturation, payroll preparation, workplace planning, flexpool
Cost:
Web Function: Yes, the webaccess module enables employees to review their schedule through the intranet or internet. They can submit wishes and requests for free days, exchange shifts and enter hour declarations. This enables your planner to be operational 24 hours a day, 7 days a week.
Demo Available: Yes

Site is not clear, but no mentioning of automated scheduling. Just aid in scheduling. See Section 2.2 for more details on Ortec Harmony.
Figure A.6: Screenshot of the Rostar software.

Website: [http://www.paralax.nl](http://www.paralax.nl)

Size:

Customers: Efteling


Software: Rostar Flex / Rostar CAS

Key focus: Yes

Extra functions: Checks, Management information, Integration with wordprocessors / databases

Cost:

Web Function: Yes, Network multi user / planner is optional.

Demo Available:

Rostar Flex and Rostar CAS (both packages from Paralax) are Computer Aided Planning. The automated planning can be added as an optional module. This module will be tailor made for the company. No guarantees about quality of result are mentioned on the website.
Primavera

Figure A.7: Graphic representation of the Primavera software.

Website: [http://www.primavera.com](http://www.primavera.com)
Size: 538 employees
Customers: AFLAC Project Management Office (PMO) Program, Dublin Light Rail, EDS Enterprise Deployment, Guardian Life Insurance PMO Program, Hartsfield-Jackson Atlanta International Airport, Hong Kong Airport, INVENSYS at Mile High Stadium (Denver, CO), Liberty Bell Center, London Underground, Los Angeles International Airport, Oresund bridge (linking Denmark and Sweden), Pentagon Phoenix Project, Philadelphia Airport, Porsche Boxster, San Francisco Airport, Shell Oil Exploration Program, and Three Gorges Dam (China)
Software: Custom made adaptation
Key focus: Project, resource and portfolio management
Extra functions:
Cost:
Web Function:
Demo Available: No
Quintiq

http://www.quintiq.com

Focusses on very large planning problems. Not the size of planning problems relevant for this search.

SAP APO

http://help.sap.com/saphelp_apo

SP-Expert

http://www.sp-expert.nl
There are four companies who offer a planning solution based on SP-Expert
SP-Expert: Intus

Figure A.8: Screenshot of the Intus version of SP-Expert.

Website: [http://www.intus.nl](http://www.intus.nl)

Size: 

Customers: RSC Rotterdam, Holland Casino, Rail4Chem, Justice department, Zoetermeer City Hall, Bayernoil, Alcatel, BYK Chemie, P&G, Bayer, UZA, Austrian, British Airways, Adac, BFG Invest, IBM, Kraft Jacobs Suchard

Branches: Industrial Activities, Transport & Logistics, Entertainment

Software: SP-Expert

Key focus: Yes

Extra functions: Time management, forecast, Time registration, workspace planning, webterminal, interfacing, management information

Cost:

Web Function: Yes

Demo Available:

Webroster is a webbased scheduling module of SP-Expert. Aided by webroster the employee can view his schedule, exchange shifts and plan holidays and free days from anywhere.

Webroster is completely integrated with SP-Expert. Changes are applied instantaneously
and are direct available for employee and planner.

Both enormous and very small groups of users can be supported by means of webroster. The application is scalable, it is possible to get an application ranging from simple schedule publication to advanced shiftbidding systems.

Webroster uses standard technology and is therefore easily integrated into the current computer system. Future support and maintenance are guaranteed.

**SP-Expert: Planning IT**

![Figure A.9: The optimization overview of the SP-Expert software.](http://www.planningit.nl)

Website: [http://www.planningit.nl](http://www.planningit.nl)
Size:
Customers:
Branches: Healthcare, Public Sector, Professional Services, Retail, Construction, Banking, Insurance, Hotels & Amusement Parks
Software: SP-Expert
Key focus:
Extra functions:
Cost:
Web Function: Intranet / internet environment enabled
Demo Available:

SP Expert uses a genetic algorithm to automatically schedule tasks.
**SP-Expert: Planpoint**

![Screenshot of the 'Rooster Direct Pakket' software.](http://www.planpoint.nl)

Figure A.10: Screenshot of the 'Rooster Direct Pakket' software.

Website: [http://www.planpoint.nl](http://www.planpoint.nl)

Size:

Customers:

Branches:

Software: Rooster direct pakket (powered by SP-Expert)

Key focus:

Extra functions:

Cost: Total (one time costs) for a contract of 50 employees and 5 days implementation is €3,520,- excluding tax.

Web Function: Semi, the planning software can be run on a remote server (or buy licence for inhouse use of SP-Expert)

Demo Available:

**SP-Expert: spen**

[No picture available]
http://www.spen.nl/
Has merged with Planning IT

Symagic

[No picture available]

Website: [http://www.symagic.nl](http://www.symagic.nl)
Size:
Customers: TNO, Getronics Car Services, Yves Rocher, Christal Teleservice
Branches: Callcenter, ICT Organisations, Retail, Financial services, Homeshopping, Retail, Healthcare, Service organisations
Software: The Symagic workforce management solution
Key focus: No
Extra functions:
Cost:
Web Function: Semi, run the software on remote server, no remote login for employees
Demo Available:
No mentioning of automated planning, just planning aid.

Synusflex

[No picture available]

http://www.synusflex.com/
Not relevant for this search, focus is on inter company planning to reduce over and under-capacity.

Unit4agresso

[No picture available]
Website: [http://www.unit4agresso.nl](http://www.unit4agresso.nl)
Size:
Customers:
Branches: Concrete Industry, Chemical Industry, Woodcraft Industry, etc.
Software: Unit 4 Multivers Productie
Key focus: No
Extra functions:
Cost:
Web Function:
Demo Available: No

Website is not clear, but most likely the software does not have automated planning. Also their software is very branch specific.
Appendix B

AIMMS libraries function reference

B.1 Flow optimization

TwoLayerFlow

The function TwoLayerFlow solves a flow problem of two layers and returns the resulting flow as a schedule that denotes for every employee (Layer1) and every Timeslot what task (Layer2) is assigned.

TwoLayerFlow(
    Layer1, (input) Set
    Layer2, (input) Set
    Timeslots, (input) Set
    ToLayer1, (input) Parameter
    Layer1ToLayer2, (input) Parameter
    FromLayer2, (input) Parameter
    Result, (output) Element parameter
)

Arguments

Layer1: Set of resources that form the first layer of the flowgraph (in the cases of Section 3.1 and 3.2 employees).

Layer2: Set of resources that form the second layer of the flowgraph (in the cases of Section 3.1 and 3.2 tasks).

Timeslots: Set of timeslots for which the schedule has to be found, a flowgraph is solved for each of the elements in the set of Timeslots.
**ToLayer1:** Parameter (indexed over Layer1) that contains a 1 if the source node and the resource of Layer1 should be connected and a 0 otherwise (in the cases of Section 3.1 and 3.2 employee availability).

**Layer1ToLayer2:** Parameter (indexed over Layer1 and Layer2) that contains a 1 if the resource of Layer1 should be connected to the resource of Layer2 and a 0 otherwise (in the cases of Section 3.1 and 3.2 employee, task competence).

**FromLayer2:** Parameter (indexed over Layer2) that contains the capacity of the Layer2 to sink connections (in the cases of Section 3.1 and 3.2 the number of times a task has to be performed in this Timeslot).

**Result:** Element parameter (indexed over the sets Layer1 and Timeslots with range Layer2) that denotes the results of the flowgraph (in the cases of Section 3.1 and 3.2 the resulting schedule).

**ThreeLayerFlow**

The function

```
ThreeLayerFlow(
    Layer1,  (input) Set
    Layer2,  (input) Set
    Layer3,  (input) Set
    Timeslots,  (input) Set
    ToLayer1,  (input) Parameter
    Layer1ToLayer2,  (input) Parameter
    Layer2ToLayer3,  (input) Parameter
    FromLayer3,  (input) Parameter
    Result,  (output) Element parameter
)
```

**Arguments**

**Layer1:** Set of resources that form the first layer of the flowgraph (in the cases of Section 3.3 students).

**Layer2:** Set of resources that form the second layer of the flowgraph (in the cases of Section 3.3 student-timeslots).

**Layer3:** Set of resources that form the third layer of the flowgraph (in the cases of Section 3.3 courses).

**Timeslots:** Set of timeslots for which the schedule has to be found, a flowgraph is solved for each of the elements in the set of Timeslots.
**ToLayer1:** Parameter (indexed over Layer1) that contains a 1 if the source node and the resource of Layer1 should be connected and a 0 otherwise (in the cases of Section 3.1 and 3.2 employee availability).

**Layer1ToLayer2:** Parameter (indexed over Layer1 and Layer2) that contains a 1 if the resource of Layer1 should be connected to the resource of Layer2 and a 0 otherwise (in the cases of Section 3.3 it is always 1).

**Layer2ToLayer3:** Parameter (indexed over Layer2 and Layer3) that contains a 1 if the resource of Layer2 should be connected to the resource of Layer3 and a 0 otherwise (in the cases of Section 3.3 a 1 when the student wants the course and the course is given in this timeslot).

**FromLayer3:** Parameter (indexed over Layer2) that contains the capacity of the Layer2 to sink connections (in the cases of Section 3.1 and 3.2 the number of times a task has to be performed in this Timeslot).

**Result:** Element parameter (indexed over te sets Layer1 and Timeslots with range Layer2) that denotes the results of the flowgraph (in the cases of Section 3.1 and 3.2 the resulting schedule).

**LagrangeTwoLayerFlow**

The function *LagrangeTwoLayerFlow* solves a flow problem with possibly follow up appointments by means of Lagrange relaxation as described in Chapter 4.

LagrangeTwoLayerFlow(  )

**Arguments**

The LagrangeTwoLayerFlow currently contains no parameters since it takes all the input from parameters present at the interface section of the administrative case library. A recommendation is to rewrite this function such that all the information is given as input instead of taken from the interface section.

**DllSolveMinCostFlow**

The function *DllSolveMinCostFlow* is only used internally in the AIMMS flow module. It is the link between the AIMMS parameters and the Goldberg flowsolver DLL.

85
Naam( 
    Nodes, (input) Set 
    NodeSupply, (input) Parameter 
    Arcs, (input) Set 
    FromNodeOfArc, (input) Element parameter 
    ToNodeOfArc, (input) Element parameter 
    ArcCost, (input) Parameter 
    ArcMinCapacity, (input) Parameter 
    ArcMaxCapacity, (input) Parameter 
    ArcOptimalFlow, (output) Parameter 
    ErrorMessage (output) String 
)

Arguments

Nodes: Set of nodes in the flow graph.

NodeSupply: Parameter (indexed over the nodes) that denotes the supply/demand of each node.

Arcs: Set of arcs in the flow graph.

FromNodeOfArc: Element parameter (indexed over the arcs with range nodes) that denotes the start node of the arc.

ToNodeOfArc: Element parameter (indexed over the arcs with range nodes) that denotes the ending node of the arc.

ArcCost: Parameter (indexed over the arcs) that denotes the costs of sending one unit of flow over the arc.

ArcMinCapacity: Parameter (indexed over the arcs) that denotes the minimum capacity of the arc.

ArcMaxCapacity: Parameter (indexed over the arcs) that denotes the maximum capacity of the arc.

ArcOptimalFlow: Parameter (indexed over the arcs) that denotes the optimal amount of flow over the arc.

ErrorMessage: String that contains an error message when something goes wrong with the Goldberg flow library.
Bottleneck

The function Bottleneck is only used internally in the AIMMS flow module by the TwoLayerFlow function. It can be started (through a dialog) when the flow is infeasible to find out the set of tasks that does not have enough competent employees to fill them.

Bottleneck(
    )

Arguments

The bottleneck function uses no external input and gets all required information from parameters present within the flow library itself.

LagrangePostFix

The function LagrangePostFix is only used internally in the AIMMS flow module by the LagrangeTwoLayerFlow function. It is started after each Lagrange iteration step, to try and post-fix (as described in paragraph 4.8) the Lagrange relaxed solution in order to find a feasible schedule.

LagrangePostFix(
    )

Arguments

The LagrangePostFix function uses no external input and gets all required information from parameters present within the Lagrange section of the flow library itself.

B.2 Gantt chart

Erase Gantt Chart

The function EraseGanttChart empties the current data of the Gantt chart object. It erases the sets and the parameters shown.

EraseGanttChart(
    )
Display Gantt Chart

The function \textit{DisplayGanttChart} shows the given schedule in a Gantt chart object.

DisplayGanttChart(
  Resources, (input) Set
  AssignedResources, (input) Set
  Timeslots, (input) Set
  Schedule, (input) Element parameter
  Availability, (input) Parameter
  Weekly, (optional) Binary variable
  ShiftsPerDay (optional) Integer
)

\textit{Arguments}

\textbf{Resources:} Set of employees in the schedule.

\textbf{AssignedResources:} Set of scheduled tasks.

\textbf{Timeslots:} Set of timeslots.

\textbf{Schedule:} Element parameter (indexed over the sets Resources and Timeslots) with range AssignedResources that denotes which task is done by an employee on each timeslot.

\textbf{Availability:} Parameter (indexed over the sets Resources and Timeslots) that has a 1 if the employee is available and a 0 otherwise.

\textbf{Continuous:} A binary variable that denotes if it is a weekly schedule (1) or not (0). If a 1 is chosen the Continuous Gantt chart object is used, if a 0 is chosen the discrete Gantt chart object is used instead.

\textbf{ShiftsPerDay:} An integer parameter that is only used in case of a continuous Gantt chart, where it denotes the number of timeslots in a day.

\textbf{B.3 Case files}

\textbf{ReadDatabase}

The function \textit{ReadDatabase} is used to read the case data from the Microsoft Access database (through ODBC).

ReadDatabase(  )
Arguments

The ReadDatabase function fills the sets and parameters present within the case file and thus needs no input. The ReadDatabase function is custom made for every case.

WriteDatabase

The function WriteDatabase is used to write the created schedule to the case data in the Microsoft Access database (through ODBC).

WriteDatabase()

Arguments

The WriteDatabase function writes the resulting schedule (present in the case library) to the database and needs no input. The WriteDatabase function is custom made for every case.

ComputeSchedule

The function ComputeSchedule is used to set all the parameters right that are needed by the TwoLayerFlow, and ThreeLayerFlow, respectively functions of the flow module. Then the Two or ThreeLayerFlow function is called from the flow library. When the flow library is finished the resulting schedule is displayed in a Gantt chart object by means of the DisplayGanttChart function of the Gantt library.

ComputeSchedule()

Arguments

The ComputeSchedule function is the overhead wrapper that calls the functions from other libraries. It also creates the right parameters for the flow library and Gantt library functions. All these parameters are created in the case file and are not used as input/output for the computeschedule library.
ExcelRead

The function ExcelRead is used to read the case data from the Microsoft Excel case files.

ExcelRead()

Arguments

The ExcelRead function fills the sets and parameters present within the case file and thus needs no input. The ExcelRead function is custom made for every case.

ExcelWrite

The function ExcelWrite is used to write the created schedule to the case data in the Microsoft Excel case files.

ExcelWrite()

Arguments

The ExcelWrite function writes the resulting schedule (present in the case library) to the Excel case file and needs no input. The ExcelWrite function is custom made for every case.

B.4 Administrative company case files

SolveMIP_CPLEX

The function SolveMIP_CPLEX solves the Mixed Integer Programming definition of the case by means of the CPLEX MIP solver. Then the MIP result is used to create the element parameter schedule. Then DisplayGanttChart is called to give the resulting schedule in a Gantt chart object.

SolveMIP_CPLEX()

Arguments

The solveMIP_CPLEX invokes CPLEX to solve the MIP that was made for the Administrative problem. Since it is not made generic an implementation with arguments has not been made.
SolveMIP_XA

The function SolveMIP_XA solves the Mixed Integer Programming definition of the case by means of the XA MIP solver. Then the MIP result is used to create the element parameter schedule. Then DisplayGanttChart is called to give the resulting schedule in a Gantt chart object.

SolveMIP_XA(
    
)

Arguments

The solveMIP_XA invokes XA to solve the MIP that was made for the Administrative problem. Since it is not made generic an implementation with arguments has not been made.

SolveGreedy

The function SolveGreedy uses the greedy method to fix the follow up appointments and then uses the flow library to fill the rest of the schedule (as described in Section 3.1.5). Then the results are forwarded to the DisplayGanttChart to give the resulting schedule in a Gantt chart object.

SolveGreedy(
    
)

Arguments

The solveGreedy solves the Administrative problem with follow up. Since it is not made generic an implementation with arguments has not been made.

SolveLagrange

The function SolveLagrange uses the Lagrange and flow method of Chapter 4 to find a feasible schedule that includes the follow up tasks. Then the results are forwarded to the DisplayGanttChart to give the resulting schedule in a Gantt chart object.

SolveLagrange(
    
)

Arguments

The solveLagrange solves the Administrative problem with follow up. Since it is not made generic an implementation with arguments has not been made.