Examensarbete

Utilizing problem specific structures in branch and bound methods for manpower planning

Björn Morén

LITH-MAT-EX-2012/10-SE
Utilizing problem specific structures in branch and bound methods for manpower planning

Division of Optimization, Linköpings Universitet

Björn Morén

LiTH-MAT-EX–2012/10–SE

Examensarbete: 30 hp

Level: A

Supervisor: Fredrik Altenstedt.
Jeppesen, Göteborg

Supervisor: Åsa Holm.
Division of Optimization, Linköpings Universitet

Examiner: Torbjörn Larsson.
Division of Optimization, Linköpings Universitet

Linköping: August 2012
Abstract

This thesis is about solving the manpower planning problem concerning staffing and transitioning of pilots. The objective of the planning is to have enough pilots to satisfy the demand while minimizing the cost. The main decisions to take are how many pilots to hire, which pilots to train and which courses to schedule. The planning problems that arise are both large and difficult which makes it important to use efficient solution methods. Seniority rules between pairs of pilots are the most complicating factor.

A major part in the solution process is the solving of mixed integer programs. The emphasis in the thesis is to develop and test adaptations of the branch and bound algorithm to solve mixed integer programs faster. One of these is a branching principle that takes a problem specific structure into account. A graph of implications is constructed from the seniority rules and this graph is then used to estimate the impact of each branching candidate. The implemented methods outperform the software XPRESS on some instances, while for most instances the performance is comparable.

Keywords: Manpower planning, Airlines, Optimization, Branch and bound, Branching methods.
Acknowledgements

I would like to thank Jeppesen for giving me this opportunity. Fredrik Altenstedt and Tomas Gustafsson have been a great help during the work. Björn Thalén and many others at Jeppesen did also help me to improve this thesis.

I would also like to thank Åsa Holm and Torbjörn Larsson at Linköping University, as well as Mikael Call.

My family and classmates have been a great support during all years.

Finally I would like to thank Boris Carlsson for endless motivating discussions and whose credit it is that I choose this path.
# Contents

1 Introduction .................................................. 1
   1.1 Outline .................................................. 1

2 Manpower Planning ........................................ 3
   2.1 Positions ................................................. 3
   2.2 Courses .................................................. 3
   2.3 Seniority ................................................ 4
   2.4 Bidding Systems ......................................... 5
   2.5 Training Resources ....................................... 5
   2.6 Optimization Problem ................................... 6
       2.6.1 Example ............................................. 6

3 Mathematical Model .......................................... 9
   3.1 Sets ...................................................... 9
   3.2 Parameters .............................................. 9
   3.3 Variables ............................................... 10
   3.4 Objective function ...................................... 11
   3.5 Constraints ............................................. 11
   3.6 Shortage of Manpower ................................... 12
   3.7 Preferences .............................................. 12
   3.8 Seniority ............................................... 12
       3.8.1 Example, part 1 .................................... 13
       3.8.2 Strengthened Inequalities ......................... 14
       3.8.3 Example, part 2 .................................... 16

4 Branch and Bound ............................................ 17
   4.1 Introduction ............................................. 17
       4.1.1 Land-Doig-Dakins Algorithm ...................... 18
   4.2 Pseudo-cost ............................................. 19
       4.2.1 Initialization .................................... 20
   4.3 Node Selection .......................................... 20
       4.3.1 Backtracking and Branch Selection ............... 20
       4.3.2 Common Rules .................................... 20
       4.3.3 Distribution Node Selection ....................... 21
   4.4 Branching ................................................. 22
       4.4.1 Pseudo-cost Branching ......................... 22
       4.4.2 Strong Branching ................................ 23
       4.4.3 Reliability Branching ......................... 23

Morén, 2012. ix
Chapter 1

Introduction

The whole planning process that airlines face is tremendously difficult. To be able to deal with it, it is necessary to break the problem apart into smaller, but still hard, subproblems. With the increased potential and capability of optimization as a tool for solving these problems, the usage of optimization techniques has also been growing.

This thesis project has been made on behalf of the company Jeppesen, which has provided a supervisor, an office as well as a large amount of computational resources. Jeppesen handles a large range of services for flight information and planning. It has offices worldwide with many of the world’s airlines as customers. The office in Gothenburg is one of the places where products involving planning and optimization are developed. Crew scheduling was an early area for optimization research and with great success, which has spread the use of optimization to other areas, with manpower planning being one of them.

In manpower planning the problem is how to supply enough pilots of each category during a given time period. In this thesis the time period stretches from one to four years. Questions to be answered are which pilots to train and when to train them, as well as if to hire new pilots pilots and also how to distribute vacations. This problem can be modeled as a mixed integer program and a common way to solve such mixed integer programs is to use a branch and bound algorithm. Currently, the software XPRESS is used to solve the branch and bound problems that arise.

At Jeppesen the problem is solved in a process involving several steps. First a starting solution is constructed which is followed by an iterative scheme to improve the starting solution. One major component in this scheme is the solving of mixed integer programs of various complexities.

This thesis deals with methods for improving the current branch and bound algorithm by considering the structure of this particular problem. Both branching methods and node selection methods are investigated. A comparison with XPRESS is also made.

1.1 Outline

This thesis is organized as follows. In Chapter 2 manpower planning is introduced followed by the mathematical model to be solved in Chapter 3. The
general solution method, which is branch and bound, along with important concepts are then explained in Chapter 4. In Chapter 5 the methods that have been used in this thesis are described. Finally the results are given as well as a discussion and suggestions for future research in Chapter 6 and 7, respectively.
Chapter 2
Manpower Planning

There are several important optimization problems in the airline industry today, such as fleet assignment, tail assignment, manpower planning, crew pairing, crew rostering, and recovery planning. For SAS in the year 2011 the cost of employees accounted for 34.1% of all expenses (SAS [19]). This indicates that small savings in percent can have a large effect on the economy of an airline.

The focus in this thesis is manpower planning and more specifically staffing and transitioning. For a survey of other optimization problems, see for example Grönkvist [4], Holm [5], or Thalén [12]. This chapter starts with an explanation of some important concepts and ends with a description of the manpower planning problem.

2.1 Positions

The main entity in the problem is the pilot. Each pilot has a position which consists of three attributes. These are fleet, seat, and base. The fleet is which type of aircraft (or in some cases, types of aircraft) that the pilot is eligible to fly. Seat is which role or rank the pilot has. There is a captain and at least one first officer on each flight and on some flights there are also one or more second officers. Finally each pilot has a home base at which the pilot starts and ends each trip (Yu et al. [15]). In most cases a pilot is required to have the exact position matching a flight, but an exception is so called “fly below rank” where for example a captain can fly as a first officer, or a first officer can fly as a second officer.

Usually pilots change position several times during their careers in order to move to larger fleets and to seats with higher ranks. Holm [5] gives two examples, one airline where pilots change positions on average 2.4 times during their careers and the airline SAS with approximately six changes of positions for each pilot during their careers. The main reason for pilots to change position is increased salary and responsibility (Yu et al. [15]).

2.2 Courses

To change position a pilot needs to take a course. Each course consists of one or more slots and slots in the same course might result in different positions.
One example of this is a course that consists of one slot to become a captain on a fleet and one slot to become a first officer on the same fleet. Therefore the prerequisites for a course may differ between slots. Courses where a pilot change fleet and seat are the most expensive while a course that change seat within a fleet from first officer to captain is not as expensive and time consuming. However, there are complicating aspects that determine which slot that is required for the promotion. The time it was since the pilot flew with the relevant aircraft (if any earlier experience) can for example affect the length of the course.

Courses where pilots change fleet or seat are divided into three parts. First there is a theoretical part as a preparatory step. The second step is training with a simulator that is specific for the type of aircraft that the pilot is going to operate. Training in a simulator requires one captain and one first officer. There are some training alternatives, such as training two captains at the same time (where one participant is a captain and the other takes the role of first officer in each training session) or training only one pilot together with a stand-in who is already a qualified pilot. Training two captains is a bit more expensive while using a stand-in is much more expensive. Even without considering the extra cost it is preferred to have one captain and one first officer. In the first and second steps the pilot is away from production which yields large costs. The final step is in-flight training during regular flights. Here, the pilot needs to fly with another pilot, that is qualified as instructor, but in-flight training contributes to production. If the pilot trains to become a captain, another captain (the instructor), who takes the first officer seat, is usually needed during the in-flight training.

Pilots start to receive the salary of the new position when they begin the course.

2.3 Seniority

When airlines transfer pilots to new positions the seniority is essential for which transfers (promotions) that are allowed. Seniority is based on how long the pilot has been working at the airline, but there are other factors that might affect seniority as well. Each pilot has a unique seniority number for each position he might get promoted to and promotions must be done in seniority order. In cases where a pilot is transferred to an unwanted position this is done in reversed seniority order.

The most important exception is the lock-in period. This is a period when the pilot is considered less senior than any pilot that is not locked in, when applying for a position. A lock-in period occurs after a pilot has changed fleet and/or seat, and usually a pilot is locked in for about three to six years. Lock-in rules may differ depending on which position the pilot applies for. Some positions are considered equal from the company’s perspective, and a pilot is locked-in for a longer time with respect to a position that is comparable to the current. The main reason for lock-in periods is to make sure that a pilot does not transfer between positions too often, since this is expensive for the company.

Another possible exception from the seniority rule is pay-protection which means that a pilot that is less senior can train before the more senior pilot as long as a more senior pilot receives the new salary at the time the less senior pilot begins the training. Pay protection is not widely used in the planning phase by
2.4 Bidding Systems

Since the supply and demand for different positions change over time, with pilots retiring being the most important factor, airlines want to transfer pilots between positions continuously. Pilots regularly get the opportunity to request what positions they would like to be promoted to. This process is called system-bid-award. The two most common systems for requesting promotions are preferential bidding and un-prioritized bids. On average in Continental airlines, 15-20% of the pilots get a new position in the system-bid-awards occurring twice a year (Yu et al. [15]).

In un-prioritized bids each pilot decides upon all positions that he/she would like to be promoted to. Seniority must be taken into account in cases when a pilot is promoted to a position that another pilot is more senior to and also wants. For seniority to hold the more senior pilot must then be awarded a promotion to a position on his list in the same week or before the less senior pilot is promoted.

When using preferential bidding each pilot assigns each promotion a number, not necessarily unique, reflecting how much he wants the promotion. This gives an ordering of the potential promotions for each pilot. Seniority rules are taken into account when a pilot gets a promotion P that he is not the most senior to. Now all pilots that want this promotion and is more senior with respect to the promotion, must get a promotion such that they are at least as happy with the promotion they get as they would have been with the promotion P.

Neither courses, slots or time are included in bids in any of these bidding systems. Such aspects are introduced in the mathematical modelling of the problem.

2.5 Training Resources

For each course that is given there is a requirement of training resources consisting of instructors and simulators. Often the number of simulators is more constraining than the availability of instructors for the number of courses that is possible to give. However, the situation varies a lot between different airlines.

Simulators can either be owned by the company or they can be leased when they are needed. The portion of simulators owned varies between companies. There is a huge cost related to the simulators. To give an estimate of how expensive a full-motion simulator can be, a figure of €12 million is mentioned in [18].
2.6 Optimization Problem

The key question in manpower planning is how to cover the demand for each position efficiently or, when this is not possible, minimize the shortage. The problem is divided into time periods, since both supply and demand for pilots in different positions vary over time.

In the optimization problem the goal is to minimize a cost. In this cost the largest parts are salaries, cost for courses, cost for simulators, and shortage costs in positions. The effects of shortage in a position can be divided in two cases. Either it is possible to cover the shortage with one or more pilots working overtime or by using external pilots, or otherwise the position cannot be covered and the flight has to be cancelled, which is much more expensive. Since the cost for shortage can be hard to calculate, an estimate is used instead (Thalén [12]). Jeppesen estimates in [16] the direct cost of delaying a flight from Europe to Asia overnight to be €150 000.

The major decisions in the manpower planning problem are if a pilot should take a course and, if so, which course the pilot should take (which includes the decision of when to start training) and if any new pilots should be hired. The main potential savings are the reduction of shortage in positions and the reduction in manpower that are not needed to cover the demand. These savings are accomplished by hiring enough pilots and training the right pilots at the right time to reduce the cost of courses and salaries.

Restrictions to these decisions come mainly from the seniority rules, but the number of simulators is also often a limitation. The mathematical model and a more thorough description of the problem will be given in Chapter 3.

2.6.1 Example

To give an overview of how pilots move between positions, an example is given.

Figure 2.1 shows a typical but simplified set of possible career paths for the pilots. New recruits start at the bottom left. The arrows indicate all possible transfers and usually pilots want to transfer towards the top positions along the darker arrows. In some cases a transfer along a light arrow might occur but this is not common. The first two letters in each box are seat, where CP is captain and CO first officer. The second part of each box is the aircraft-type (in some cases more than one) that the pilot is eligible to fly.

After each transfer is a lock-in period but there is also a requirement to have worked at the company for a certain time before a promotion is possible. This time depends on the position the promotion will lead to.

Assume that for the moment there are enough pilots in all positions but it is known that a captain for A32/33 (top position) will retire the next year. If the demand is unchanged, transfers need to be done to keep enough supply for each position. The only possible transfer to position CP A32/A33 is from CP A32 and then there will be a gap in that position. To fill this gap there are two possibilities, either to promote a pilot from CP B76 or from CP A32 and then in turn the new gap must be filled, and so on. In this case it takes at least five transfers including one hired pilot to make sure that there is the same amount of pilots in each position for next year. Both which promotions to be done and at what time must be decided.
For each promotion to be valid, seniority must be respected as well as bids from all pilots. This affects both at which time a promotion is possible and sometimes which promotions that are at all feasible. Finally a plan must respect the number of training resources (simulators) available in each time period.

In this example it is cheaper to transfer a pilot from CO A32/A33 to CP A32 than from CP CJ since this pilot already has qualifications to fly with A32. It might however be the case that there is a pilot in CP CJ that is more senior with respect to CP A32 than any pilot in CO A32/A33, making the cheaper transfer unfeasible.
Chapter 3

Mathematical Model

In the first part of this chapter a complete optimization model for the manpower planning problem will be stated. Important constraints will then be explained in detail, with emphasis on seniority rules.

3.1 Sets

\( C \) is the set of all pilots.

\( K \) is the set of all courses. A course is here both a set of tasks and the time when the course starts.

\( S \) is the set of all slots.

\( J \) is the set of all resources, for example simulators.

\( H \) is the set of all positions.

\( H^2 \) is the set of pairs \((h, \bar{h})\) of positions such that it is possible to fly below rank in position \(\bar{h}\) from position \(h\).

\( T \) is the set of all time periods, for example weeks, included in the problem.

\( V(c) \) is the set of all preference values used by a pilot \(c\).

\( S^1(k) \) is the set of all slots that belong to a course \(k\).

\( S^2(c) \) is the slots eligible for pilot \(c\).

\( T_i \) for \(i = 1, \ldots, p\) is the time periods that belong to a window where vacation can be distributed freely.

3.2 Parameters

\( c_{x_{s,c}} \) is the salary for pilot \(c\) if he gets slot \(s\).

\( c_{y_k} \) is the cost to start course \(k\).
Chapter 3. Mathematical Model

$c_{s,h,t}$ is the unit cost of not having enough manpower in position $h$ in time period $t$.

cd$_s$ is the cost for having a stand-in for slot $s$.

$sen(c,s)$ is the seniority value for pilot $c$ with respect to slot $s$. The smaller the value $sen(c,s)$ is, the more senior the pilot is.

$pref(c,s)$ is the preference value that pilot $c$ has assigned slot $s$. A smaller preference value means that the slot is preferred over a slot with a higher preference value.

$v_c^-$ is the closest smaller (better) value in $V(c)$ for a pilot $c$ assuming that slot $s$ was assigned value $v$.

$v(c)$ is the smallest preference value in $V(c)$ for a pilot $c$.

$cap_{s,k}$ equals the capacity of slots $s$ in each course $k$ given.

$a_{j,k,t}$ is the amount of resource $j$ used by course $k$ in period $t$.

$rc_{j,t}$ is the maximum amount of resource $j$ that can be used in period $t$.

$a_{c,s,h,t}$ is the contribution to supply of position $h$ in period $t$ by pilot $c$ if he gets slot $s$.

$b_{h,t}$ is the demand for position $h$ in period $t$.

$dvac_{h,i}$ is the minimum amount of vacation distributed to position $h$ during the time window $T_i$.

$g_{s,h,t}$ is time away from production in position $h$ in period $t$ when using a stand-in in slot $s$.

### 3.3 Variables

$x_{s,c} \in \mathbb{B}$: Roster variable, equals one if pilot $c$ gets slot $s$ (including a slot for where the pilot does not start a course), and zero otherwise.

$\nu_{v,c} \in \mathbb{B}$: Preference variable, equal to one if pilot $c$ got a slot with preference value of at most $v$, and zero otherwise.

$y_k \in \mathbb{Z}^+$: Number of courses $k$ to start.

$d_s \in \mathbb{Z}^+$: Number of stand-ins in slot $s$.

$s_{h,t} \in \mathbb{R}^+$: Shortage variable for position $h$ in period $t$.

$w_{h,h,t} \in \mathbb{R}^+$: Fly below rank variable, which is positive if any work hours are moved from position $h$ to $h$.

$vach_{h,t} \in \mathbb{R}^+$: Vacation distributed in time period $t$ for position $h$. 
3.4 Objective function

\[ \min z = \sum_{c \in C} \sum_{s \in S^2(c)} c x_{s,c} + \sum_{k \in K} c y_k y_k + \sum_{h \in H} \sum_{t \in T} c s_{h,t} s_{h,t} + \sum_{s \in S} c d_s d_s \]

The first part of the objective function is the salaries of the pilots. The following parts are costs for courses, salaries and stand-ins.

3.5 Constraints

Each pilot is assigned a slot:

\[ \sum_{s \in S^2(c)} x_{s,c} = 1, \ c \in C \] (3.1)

Seniority rules:

\[ \nu_{pref(c_1,s),c_1} - x_{s,c_2} \geq 0, \ c_1,c_2,s : \sen(c_1,s) < \sen(c_2,s) \] (3.2)

Keep track of preference variables:

\[ \nu_{v,(c),c} = \sum_{s \in S^2(c):pref(c,s) = v} x_{s,c} = 0, \ c \in C \]
\[ \nu_{v,(c),c} - \sum_{s \in S^2(c):pref(c,s) = v} x_{s,c} = 0, \ v \in V(c) \setminus \{\bar{v}(c)\}, c \in C \] (3.3)

Start the right amount of courses:

\[ \text{cap}_{s,k} y_k - \sum_{c \in C} x_{s,c} = 0, \ k \in K, s \in S^1(k) \] (3.4)

Resources are limited:

\[ \sum_{k \in K} a_{j,k,t} y_k \leq r_{c,j,t}, \ j \in J, t \in T \] (3.5)

Track shortage of manpower:

\[ \sum_{c \in C} \sum_{s \in S^2(c)} a_{c,s,h,t} x_{s,c} + \\
+ \sum_{h: (h,h) \in H^2} w_{h,h,t} - \sum_{h: (h,h) \in H^2} w_{h,h,t} - \\
- g_{s,h,t} d_s - vac_{h,t} + s_{h,t} \geq b_{h,t}, \ h \in H, t \in T \] (3.6)

Distribute enough vacation:

\[ \sum_{t \in T_i} vac_{h,t} \geq d_{vac,h,i}, \ h \in H, i = 1, \ldots, p \] (3.7)

The first set of constraints, 3.1 makes sure that each pilot gets a slot (including the possibility to not start a course). Constraints 3.4 and 3.5 define course variables and limit the number of courses started each time period in terms of available resources (simulators). Constraint 3.7 forces vacation to be distributed in time periods within a window. The other constraints will be explained in the following sections.
3.6 Shortage of Manpower

Constraint 3.6 is used to track shortage of manpower if there is any. The total supply on the left hand side consists of several components. The first term is how much each pilot contributes to the supply depending on what course he takes. It is possible that a pilot contributes to one position in one week and after some training contributes to another position in a later week. The first term involving \( w_{h,h,t} \) takes into account all possibilities to get supply from another position by pilots flying below rank, while the second term involving \( w_{h,h,t} \) is all supply moved to another position with fly below rank. Next there is a reduction in supply if a stand-in is used in a slot. This stand-in is taken from the position that the slot leads to. Supply is also reduced in each period where vacation was distributed. Now if the total supply is smaller than the demand this must be compensated using variable, \( s_{h,t} \) which is the shortage of manpower. This can be regarded as an expensive form of manpower.

The variable \( w_{h,h,t} \) describing fly below rank is one example of how manpower can be distributed among positions. The model also supports other constructions that may arise in practice. One case is where a position permits a pilot to fly with two different fleets.

3.7 Preferences

The recursively constructed set of constraints 3.3 is used to define values of the preference variable \( \nu \). The variable \( \nu_{v,c} \) equals one if pilot \( c \) got a slot that he gives a value that is not higher than \( v \) (the smaller the preference value is the more preferred is the slot).

It is important to note how preference variables for a pilot are related through constraints 3.3. If a preference variable \( \nu_{v,c} \) is equal to one this implies that all other preference variables which correspond to preference values that are higher (worse) than \( v \) are also equal to one. The effect of \( \nu_{v,c} \) being equal to zero is that all other preference variables \( \nu_{v',c} \) with \( v' \leq v \) must also be equal to zero. Thus the preference variables are zero until a point where the value changes to one and then all the following preference variables are also equal to one.

It is possible that two different slots are given the same preference value. One natural example of this is where there are two courses to become a captain, one with two slots to become a captain and the other course with one slot to become a captain and one slot to become a first officer.

The preference variables are used to express the seniority rules in a simpler way and they also reduce the density of the constraint matrix.

3.8 Seniority

There is a constraint of type 3.2 in each case that a pilot \( c_1 \) is more senior than a pilot \( c_2 \) with respect to a slot \( s \). It states that if pilot \( c_2 \) gets slot \( s \), the pilot \( c_1 \) must get a slot that he/she considers as at least as good as \( s \).

After an example it will be shown how the seniority constraints from 3.2 can be strengthened.
3.8. Seniority

3.8.1 Example, part 1

Figure 3.1 shows a subset of all the possible career paths introduced in Figure 2.1. In the first part of this example, preference variables for two pilots will be introduced and then the corresponding seniority constraints will be set up.

![Career Paths Diagram]

Figure 3.1: A subset of all career paths from position CO A32.

Assume that there are two pilots in the position of CO A32, \( p \) and \( c \). For both pilots there are three positions that they are eligible to transfer to. These are CO B76, CO A33, and CP CJ. The option to stay in the current position is also possible, giving a total of four slots. In Table 3.1 the pilots' evaluations of each promotion are listed. Note that the slot with the smallest preference value is the most preferred.

<table>
<thead>
<tr>
<th>( s_3 = \text{CP CJ} )</th>
<th>( s_2 = \text{CO A33} )</th>
<th>( s_1 = \text{CO B76} )</th>
<th>( s_0 = \text{stay at CO A32} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p )</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>( c )</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.1: Preference values for two pilots and four available slots.

For pilot \( p \), four preference variables are used to describe the order of the promotions. Preference variable \( \nu_{1,p} \) is 1 if pilot \( p \) gets promoted to position CP CJ. Next, \( \nu_{2,p} \) is 1 if he/she gets a promotion that is preferred at least as much as the one to CO A33. This includes a promotion to CP CJ. The third preference variable, \( \nu_{3,p} \), is 1 if pilot \( p \) gets any of the promotions. Finally the fourth preference variable, \( \nu_{4,p} \), can be omitted since it is always equal to one due to the single roster constraints 3.1.

Using the same arguments, it is enough with three preference variables for pilot \( c \). One preference variable describing if he/she gets promoted to CP CJ, one if he/she gets a promotion to CO B76 or CP CJ, and a third one if pilot \( p \) gets any of the available promotions.
Using the preference variables it is possible to construct the seniority constraints. Assume that pilot \( p \) is more senior than pilot \( c \) with respect to all slots (promotions). Then there are three forbidden scenarios.

1. Pilot \( c \) gets promoted to CP CJ while pilot \( p \) does not get a promotion to CP CJ.

2. Pilot \( c \) gets promoted to CO A33 but pilot \( p \) does not get any of the promotions to CP CJ or CO A33.

3. Pilot \( c \) gets promoted to CO B76 but pilot \( p \) stays at the current position (CO A32).

This corresponds to the following three inequalities that must be included in the model.

\[
\nu_{1,p} - x_{s_3,c} \geq 0, \\
\nu_{2,p} - x_{s_2,c} \geq 0, \\
\nu_{3,p} - x_{s_1,c} \geq 0.
\]

### 3.8.2 Strengthened Inequalities

Although constraints 3.2 are enough to construct a valid mathematical modeling of the seniority restrictions, it is possible to strengthen in order to obtain a formulation with a stronger LP-relaxation.

**Theorem 1.** Consider the case of the two inequalities

\[
\nu_{\text{pref}}(c_1,s_1),c_1 - x_{s_1,c_2} \geq 0, \\
\nu_{\text{pref}}(c_1,s_2),c_1 - x_{s_2,c_2} \geq 0
\]

in the problem formulation where \( c_1 \) is more senior than \( c_2 \) with respect to both \( s_1 \) and \( s_2 \) and \( c_1 \) prefers slot \( s_2 \) over \( s_1 \). The formulation can be strengthened to

\[
\nu_{\text{pref}}(c_1,s_1),c_1 - x_{s_2,c_2} - x_{s_1,c_2} \geq 0, \\
\nu_{\text{pref}}(c_1,s_2),c_1 - x_{s_2,c_2} \geq 0.
\]

**Proof.** First note that the feasible set with the new constraints is contained in the feasible set with the original constraints. What needs to be shown is therefore that any feasible integer point satisfying the original constraints also satisfies the new constraints.

Add the two inequalities to get

\[
\nu_{\text{pref}}(c_1,s_1),c_1 + \nu_{\text{pref}}(c_1,s_2),c_1 - x_{s_2,c_2} - x_{s_1,c_2} \geq 0.
\]

Let \( S = \{ s \in S(c_1) : \text{pref}(c_1,s_2) < \text{pref}(c_1,s) \leq \text{pref}(c_1,s_1) \} \). Expressed in words, the set \( S \) contains all slots that are both less preferred than \( s_2 \) and considered at least as good as \( s_1 \). Since, from 3.3, \( \nu_{\text{pref}}(c_1,s_1),c_1 = \nu_{\text{pref}}(c_1,s_2),c_1 + \sum_{s \in S} x_{s,c_1} \), the inequality can be written

\[
2\nu_{\text{pref}}(c_1,s_2),c_1 + \sum_{s \in S} x_{s,c_1} - x_{s_2,c_2} - x_{s_1,c_2} \geq 0.
\]
3.8. Seniority

Now dividing by two and rounding up all positive terms gives
\[ \nu_{\text{pref}}(c_1, s_2, c_1) + \sum_{s \in S} x_{s, c_1} - \frac{1}{2} x_{s_2, c_2} - \frac{1}{2} x_{s_1, c_2} \geq 0. \]  
(3.8)
The sum \( x_{s_2, c_2} + x_{s_1, c_2} \) can be either 0 or 1, because of constraints 3.1, giving two cases.

- If \( x_{s_2, c_2} + x_{s_1, c_2} = 0 \), then
  \[ \nu_{\text{pref}}(c_1, s_2, c_1) + \sum_{s \in S} x_{s, c_1} - x_{s_2, c_2} - x_{s_1, c_2} \geq 0 \]  
  (3.9)
is valid since \( \nu, x \geq 0 \).

- If \( x_{s_2, c_2} + x_{s_1, c_2} = 1 \), then for inequality 3.8 to hold, one of the positive terms must be strictly positive and hence equal to one and thus 3.9 holds.

Hence the constraint is valid in both cases and replacing \( \nu_{\text{pref}}(c_1, s_2, c_1) + \sum_{s \in S} x_{s, c_1} \) with \( \nu_{\text{pref}}(c_1, s_1, c_1) \) yields the constraint in standard form,
\[ \nu_{\text{pref}}(c_1, s_1, c_1) - x_{s_2, c_2} - x_{s_1, c_2} \geq 0, \]
which is then also valid. Thus any feasible integer point satisfying the original constraints also satisfies the new constraints.

This theorem can then be used repeatedly to strengthen the set of constraints 3.2. For cases where \( \text{sen}(c_1, \bar{s}) < \text{sen}(c_2, \bar{s}) \), sets \( R(c_1, c_2, \bar{s}) \) and \( R^+(c_1, c_2, \bar{s}) \) are defined as
\[ R(c_1, c_2, \bar{s}) = \{ s \in S : \text{pref}(c_1, s) \leq \text{pref}(c_1, \bar{s}) \} \]
\[ R^+(c_1, c_2, \bar{s}) = \{ s \in R(c_1, c_2, \bar{s}) : \text{sen}(c_1, s) < \text{sen}(c_2, s) \}. \]
The set \( R(c_1, c_2, \bar{s}) \) contains all slots that pilot \( c_1 \) regards as at least as good as \( \bar{s} \). The set \( R^+(c_1, c_2, \bar{s}) \) is then all of these slots for which \( c_1 \) is more senior than \( c_2 \).

This gives the new formulation of the seniority constraints,
\[ \nu_{\text{pref}}(c_1, s_1, c_1) - \sum_{s \in R^+(c_1, c_2, \bar{s})} x_{s, c_2} \geq 0, \quad c_1, c_2, \bar{s} : \text{sen}(c_1, \bar{s}) < \text{sen}(c_2, \bar{s}), \]
(3.10)
which replace constraints 3.2.

In some cases \( \sum_{s \in R^+(c_1, c_2, \bar{s})} x_{s, c_2} \) can be replaced by \( \nu_{\text{pref}}(c_2, \bar{s}, c_2) \). One of these cases is when the seniority relation between two pilots \( c_1 \) and \( c_2 \) is the same for all slots (which is not unusual) and the two pilots prefer slots in the same order. This allows constraints 3.10 to be rewritten as
\[ \nu_{\text{pref}}(c_1, s_1, c_1) - \nu_{\text{pref}}(c_2, s_2, c_2) \geq 0, \]  
(3.11)
where \( c_1 \) is more senior than \( c_2 \). Even if it is not possible to substitute the whole sum
\[ \sum_{s \in R^+(c_1, c_2, \bar{s})} x_{s, c_2} \]
with a preference variable it is often possible to replace a subset of the roster variables with a preference variable.
3.8.3 Example, part 2

The previous example will now be wrapped up and the constraints presented in the strengthened form. Initially the constraints are set up in the form of seniority constraints in 3.2.

\[ \nu_{1,p} - x_{s3,c} \geq 0 \]
\[ \nu_{2,p} - x_{s2,c} \geq 0 \]
\[ \nu_{3,p} - x_{s1,c} \geq 0 \]

By using Theorem 1 the first two inequalities can be combined to give

\[ \nu_{2,p} - x_{s3,c} - x_{s2,c} \geq 0. \]

This new inequality can now be combined with the third inequality using Theorem 1, giving

\[ \nu_{3,p} - x_{s3,c} - x_{s2,c} - x_{s1,c} \geq 0. \]

In the set of strengthened inequalities

\[ \nu_{1,p} = x_{s3,c} \geq 0, \]
\[ \nu_{2,p} = x_{s3,c} - x_{s2,c} \geq 0, \]
\[ \nu_{3,p} = x_{s3,c} - x_{s2,c} - x_{s1,c} \geq 0, \]

some substitutions can be done. Preference variable \( \nu_{1,c} \) can replace \( x_{s3,c} \) and preference variable \( \nu_{3,c} \) can replace the sum \( x_{s3,c} + x_{s2,c} + x_{s1,c} \). This gives the following final form of the seniority constraints.

\[ \nu_{1,p} - \nu_{1,c} \geq 0 \]
\[ \nu_{2,p} - \nu_{1,c} - x_{s2,c} \geq 0 \]
\[ \nu_{3,p} - \nu_{3,c} \geq 0 \]
Chapter 4

Branch and Bound

One of the main methods for solving mixed integer programs is branch and bound. In this chapter the concept of branch and bound will be introduced as well as the general algorithm. Some of the most recent development in the field will also be reviewed.

4.1 Introduction

This introduction is based on Wolsey [14]. Assume that we have a mixed integer program (MIP) of the following form.

\[
\begin{align*}
\text{minimize} & \quad z = cx \\
\text{subject to} & \quad Ax \geq b \\
& \quad l \leq x \leq u \\
& \quad x_i \in \mathbb{Z}, \quad i \in I
\end{align*}
\]

The set \( I \) is a subset of all variable indices. If \( I \) contains all of the variables we have a pure integer program (IP). If the point \( x \) satisfies the first two constraints and \( x_i \) is integer for all \( i \in I \), then it will be called an integer solution.

When the problem is solved with branch and bound a relaxation is considered repeatedly. The most common type of relaxation is an LP-relaxation, in which the integer requirements are removed from the problem. Other types of relaxations can be done by removing some type of constraints, with the purpose of obtaining an easier problem to solve. For the remainder of the thesis only LP-relaxations will be considered.

Branch and bound methods involve a sequence of subproblems, organized in a tree structure, and the first step at every subproblem is to solve an LP-relaxation of the problem. This is called investigating a node of the tree. If the LP-solution is feasible and bounded and there exists some \( x_i, i \in I \), that is not integer, then the solution is not valid in the original problem. To continue, the problem is partitioned into two or more subproblems. The most common partition is to branch on a fractional variable \( x_i \) with value \( \alpha \) by setting \( x_i \leq \lfloor \alpha \rfloor \) in the down branch and \( x_i \geq \lceil \alpha \rceil \) in the up branch. In each of the branches a new node is created. The nodes that are the result of branching at node \( i \)
are called children of node \( i \), and node \( i \) is called the parent of the new nodes. Before a child has been investigated, the optimal value of its LP-solution is considered to be equal to that of the LP-solution in its parent. The nodes that are created by one or more branchings from node \( i \) are called the descendants of node \( i \). This way a tree is built up that can be seen as a systematic way to search for integer solutions. In this thesis all considered branching rules give two new nodes.

Figure 4.1 shows a node that has been investigated and the two children that were created after a binary branching variable had been selected.

![Figure 4.1: Part of a branch and bound tree](image)

Since a relaxation is solved, the objective function value is an optimistic bound that is valid for the current node and all its descendants. In the case when the LP-solution of the current node is feasible in the original problem, this solution provides a pessimistic bound to the objective function value. This bound is valid in the whole tree.

An important part of branch and bound is pruning, which occurs when there is no need to continue searching in a branch. There are three possible reasons to prune a branch.

- If an integer solution is found, the current branch is pruned and the solution is saved if it is better than the best integer solution found so far.
- If the objective function value from the LP-solution is greater than or equal to the objective function value of a known integer solution the branch can be pruned. This is because no descendant of the current node can have a better objective function value.
- At last if, there is no feasible LP-solution in the current node the whole branch is infeasible and can be pruned.

### 4.1.1 Land-Doig-Dakins Algorithm

The notations used in the pseudocode below are. \( N \) for the set of nodes that still need to be investigated, \( L \) for the set of nodes that either have been pruned or have not been investigated, \( \bar{z} \) for the pessimistic bound with corresponding point \( x^* \), \( z \) for the best optimistic bound, and \( z^L_P \) for LP-solution in node \( i \) with corresponding point \( x^i \). Node 0 is used for the root node of the tree. The following algorithm is based on Nemhauser and Wolsey [9].

1. Initialization: Set \( N = \{0\} \), \( \bar{z} = \infty \) and \( \underline{z} = -\infty \).
4.2 Pseudo-cost

2. Termination: If $N = \emptyset$, terminate. If $\bar{z} < \infty$ then $x^*$ is an optimal solution with objective function value $\bar{z}$. Otherwise there exists no feasible solution to the problem.

3. Update optimistic bound: Set $\bar{z} = \min_{j \in L} z^{LP}_j$.

4. Node selection: Choose a node $i \in N$, remove it from $N$ and solve the LP-relaxation to get $z_i^{LP}$ and $x_i$.

5. Pruning: If $z_i^{LP} = \infty$, indicating an infeasible problem, go to step 2. If the LP-solution is an integer solution and $z_i^{LP} < \bar{z}$, update $\bar{z} = z_i^{LP}$ and $x^* = x_i$ and go to step 2. If $z_i^{LP} \geq \bar{z}$ go to step 2.

6. Branching: Select a variable that is fractional in the current node and create two subproblems. Then add the subproblems to the set $N$. Go to step 3.

Other criteria for termination can be used, based on for example execution time, number of nodes investigated or relative deviation between the pessimistic and optimistic bounds.

Two steps in the algorithm involve choices that need to be specified, namely which node to investigate in step 4 and which variable to branch on in step 6. These choices have a major impact on the performance of branch and bound methods (Wojtaszek and Chinnecck [13]). Various alternatives for node selection and branching will be presented in the following sections.

4.2 Pseudo-cost

Pseudo-costs are used for both node selection and branching. It was first introduced by Benichou et al. in [3]. Originally it was suggested as a branching rule. In the commercial MIP solver XPRESS all standard branching rules use pseudo-costs (XPRESS [17]).

Pseudo-costs try to estimate the effect on the objective function value by the decision to branch on a variable $j$. There are separate pseudo-costs for the up branch $P^U_j$ and for the down branch $P^D_j$. These values are usually based on results from the tree search made so far. Let variable $x_j$ be a candidate branching variable in node $i$ (giving node $2i + 1$ in the down branch and $2i + 2$ in the up branch) with value $x_j^i$ in the LP-solution. Also set $f^i_j$ to be the fractional part $x_j^i - \lfloor x_j^i \rfloor$ and assume that the values $z_{2i+1}^{LP}$ and $z_{2i+2}^{LP}$ are calculated. Then pseudo-costs are defined as follows (Benichou et al. [3]):

$$P^D_j = \frac{z_{2i+1}^{LP} - z_i^{LP}}{f^i_j}$$

$$P^U_j = \frac{z_{2i+2}^{LP} - z_i^{LP}}{1 - f^i_j}.$$ 

Pseudo-costs calculated from different nodes are expected to be the same in the tree with only a few nodes as exception (Benichou et al. [3]). Results by
Linderoth and Savelsbergh in [7] support this view with statistical results but also observe the fact that pseudo-costs can differ significantly between nodes. They therefore suggest to use an average of observed pseudo-costs.

4.2.1 Initialization

In the root node when no branchings have been made, there is no data to use to calculate pseudo-costs. Two options for initialization are to use either problem data or a lookahead strategy. Problem data can be for example objective function coefficients. When using a lookahead strategy a few dual simplex iterations are made with an added branching constraint but without creating new nodes. Then the new objective function value is used to estimate pseudo-costs (Linderoth and Savelsbergh [7]).

Four different initialization strategies for pseudo-costs were compared by Linderoth and Savelsbergh in [7], showing the best results when using a lookahead strategy for all fractional variables for which no pseudo-costs have been calculated.

4.3 Node Selection

The purpose of node selection is either to find a good integer solution or to improve the optimistic bound to be able to prove optimality (Linderoth and Savelsbergh [7]). In this section common node selection rules will be explained as well as new suggestions from the literature.

4.3.1 Backtracking and Branch Selection

The first choice that needs to be made is whether to investigate a child of the last investigated node or to search among all active nodes for the next node to investigate. The latter choice is known as to "backtrack". An advantage with investigating a child is that the last LP-solution can be used to start from. This reduces the number of simplex iterations compared to investigating a node in another part of the search tree (Wojtaszek and Chinneck [13]).

When a node has been chosen and there are two children that has not yet been investigated, it must be decided which child to investigate. Several ways to select a branch was tested by Achterberg in [1]. The method that gave the best results was the simple rule of always selecting the up branch. This is motivated by the general property that setting a variable to one is a decision with larger impact on the solution than setting a variable to zero.

4.3.2 Common Rules

**Depth-first** A child of the node investigated latest is chosen. In cases where a node is pruned, the next node to investigate is instead the latest created active node. The aim with this node selection rule is to find an integer solution fast (Wojtaszek and Chinneck [13]).

**Most-feasible** The node that minimizes the total fractionality of all variables in the LP-solution of the parent is chosen (Wojtaszek and Chinneck [13]).
4.3. Node Selection

**Breadth-first** The active node that was created earliest is chosen. This means that all nodes on a given depth will be investigated before moving on to the next depth (Lundgren et al. [8]).

**Best-bound** Focus is on improving the optimistic bound by branching from the active node with best optimistic bound. This will avoid investigation of the nodes with an optimistic bound that is worse than the optimal objective function value. Nodes that are less deep tend to have a better optimistic bound than nodes deeper in the tree, which make best-bound more likely to investigate nodes closer to the root node first (Wojtaszek and Chinnneck [13]).

**Best-projection** The change in objective value between the root node $z_{0}^{LP}$ and an integer solution $z^{IP}$ is used, as well as the sum of fractionality in the root node, $s_0$, compared to the sum of the fractionality in node $i$, $s_i$.

$$E_i = z_i^{LP} + \left(\frac{z^{IP} - z_0^{LP}}{s_0}\right)s_i$$

A node $n$ to investigate is selected as $n = \arg\min E_i$. In cases where no integer solutions has yet been found an estimate is used instead. The quality of this estimate is important for the performance of the method (Wojtaszek and Chinnneck [13]).

**Best-estimate** Pseudo-costs are used to estimate the best integer solution that can be obtained from a node. With $Q$ as the set of fractional variables, an estimate is calculated as follows:

$$E_i = z_i^{LP} + \sum_{j \in Q} \min\{p_j^D f_j^i, p_j^U (1 - f_j^i)\}$$

The next node is then selected as the one with the lowest estimated value (Benichou et al. [3]).

### 4.3.3 Distribution Node Selection

Distribution node selection is a method that was developed by Wojtaszek and Chinnneck in [13]. The purpose is to balance the search for improved integer solutions with the search to prove optimality, based on data from the investigated nodes. The number of fractional integer variables, $c_i$, is used to measure integer infeasibility. When selecting a node, both $z_i^{LP}$ and $c_i$ are desired to be minimized, and the selection rule is constructed to take both measures into account, while not letting one value dominate the other. Since the ranges of the values $z_i^{LP}$ and $c_i$ vary between problems, they have to be weighted.

The idea of the method is that each investigated node is a sample from a data set where the variables have some probability distributions. Variables are considered to be random quantities and the sum of the variables are approximated by a normal distribution. The value $z_i^{LP}$ is seen as a weighted (by objective function coefficients) sum of the variables. For integer feasibility, each integer variable add 1 to $c_i$ if fractional and 0 to $c_i$ if integer.

Estimates of mean value and standard deviation for $z^{LP}$ and $c$ are calculated from the nodes that have been investigated. Values $F_z(z) = P(Z \leq z)$ and
$F_c(c) = P(C \leq c)$ are based on estimates of the normal cumulative distribution functions. Each node $i$ is then given a value $F_{zc}(z_{iLP}, c_i) = F_z(z_{iLP}) \times F_c(c_i)$. The node $n$ to investigate next is calculated by $n = \arg \min_i F_{zc}(z_{iLP}, c_i)$.

Example

An example from Wojtaszek and Chinneck [13] will be given to clarify the method. Four nodes have been investigated with objective function values and number of fractional variables as given in the following table. There are currently two active nodes, $n_1$ and $n_2$.

<table>
<thead>
<tr>
<th>node $i$</th>
<th>objective function value $z_{iLP}$</th>
<th>fractional variables $c_i$</th>
<th>child</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>100</td>
<td>$n_1$</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>1</td>
<td>$n_2$</td>
</tr>
</tbody>
</table>

Figure 4.2: Data from investigated nodes.

Mean value for observations of objective function value and fractional variables are 75.25 and 25.75, respectively. The standard deviation is in both cases 49.5. Table 4.3 shows calculations to choose the next node to investigate.

<table>
<thead>
<tr>
<th>active node $i$</th>
<th>$F_z(z_{iLP})$</th>
<th>$F_c(c_i)$</th>
<th>$F_{zc}(z_{iLP}, c_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_1$</td>
<td>0.067</td>
<td>0.933</td>
<td>0.062</td>
</tr>
<tr>
<td>$n_2$</td>
<td>0.691</td>
<td>0.309</td>
<td>0.213</td>
</tr>
</tbody>
</table>

Figure 4.3: Candidate nodes.

Node $n_1$ is chosen as the next node to investigate since the value $F_{zc}(z_{n_1}, c_{n_1})$ is smaller. This is motivated by the indication that among the investigated nodes more emphasis has been given to fractionality than to objective function value.

4.4 Branching

Branching is about deciding which variable or entity to branch on, with the most common idea being to branch on the most important variables first (Johnson et al. [6]). A selection of branching rules will be presented in this section. Most branching rules are based on evaluations of the objective function. One of the exceptions is a collection of branching rules based on active constraints in an LP-solution (see 4.4.5).

4.4.1 Pseudo-cost Branching

Let $P^U_j$ be the pseudo-cost for the up branch of variable $x_j$ and let $P^D_j$ be the pseudo-cost for the down branch of the same variable $x_j$. Let $f^i_j$ be the fractional part of variable $x_j$ at node $i$. In Benichou et al. [3] the variable to branch on is decided as:

$$\arg \max_j \{\min \{P^D_j f^i_j, P^U_j (1 - f^i_j)\}\}.$$
Another suggestion was given by Linderoth and Savelsbergh in [7]. With

\[ \text{score}(j) = \alpha_1 \min \{ P_j^D f_j^i, P_j^U (1 - f_j^i) \} + \alpha_2 \max \{ P_j^D f_j^i, P_j^U (1 - f_j^i) \}, \] (4.1)

branching is done on the variable with highest score. The parameters \((\alpha_1, \alpha_2)\) were set to \((2, 1)\).

XPRESS provides branching strategies based on the rule 4.1 but also of other types, for example \(\arg \max_j \{ P_j^U \times P_j^D \}\) and \(\arg \max_j \{ P_j^U + P_j^D \}\) (XPRESS [17]).

4.4.2 Strong Branching

Instead of estimating what will happen if a variable is branched on, the idea with strong branching is to calculate the change. Strong branching can be seen as a search for the best variable to branch on locally. The constraint from the branching decision is added temporarily. Then the extended problem is solved, either to optimality or partially by doing a number of dual simplex iterations from the current solution. If all of the candidate variables to branch on are examined, the method is called full strong branching. If a subset of the candidates is selected to perform strong branching on, which variables to choose must be specified. Strong branching leads to a smaller search tree but with the cost of higher computational effort in each node. Strong branching is a common look-ahead strategy used to initialize and update pseudo-costs (Achterberg et al. [2]).

4.4.3 Reliability Branching

Reliability branching is a concept that was developed by Achterberg et al. in [2] and is a combination of pseudo-cost branching and strong branching, where strong branching is used to initialize pseudo-costs and to update them during the tree search. Pseudo-costs are calculated as averages of observed pseudo-costs from strong branching. Let \(\eta_j^D\) and \(\eta_j^U\) be the number of times that strong branching has been used on a variable in down and up direction respectively. A variable is called unreliable if \(\min \{ \eta_j^D, \eta_j^U \} < \eta_{rel}\) where \(\eta_{rel}\) is a parameter set by the user.

At an initial step a score is calculated for each candidate variable. The score suggested is based on 4.1 with parameters \((\alpha_1, \alpha_2) = (5/6, 1/6)\). The candidates are then sorted in a non-increasing order in which to update the variables. Start with the variable with the highest score that is unreliable. Update the pseudo-cost for this variable using the average of all observed pseudo-costs and then update the score. Continue with the next unreliable variable in order until there is no improvement of the maximal score for \(\lambda\) consecutive updates. The value of the parameter \(\lambda\) is set by the user (Achterberg et al. [2]).

In [2] Achterberg et al. suggest the following score function, which is also the default choice in the MIP-solver SCIP [11],

\[ \text{score}(j) = \max \{ P_j^D f_j^i, \epsilon \} \cdot \max \{ P_j^U (1 - f_j^i), \epsilon \} \] (4.2)

The parameter \(\epsilon\) is set to be \(10^{-6}\).
Reliability branching is a dynamic generalization of the hybrid strong/pseudo-cost branching. In hybrid strong/pseudo-cost branching strong branching is performed at each node above a chosen depth $d$. For nodes at a level deeper than $d$ pseudo-cost branching is used. The main purpose with reliability branching is to initialize the pseudo-costs with accurate data and then to use the acquired information (Achterberg et al. [2]).

4.4.4 GUB Branching

Generalized upper bounds (GUB) or special ordered sets (SOS) are constraints of the form

$$\sum_{j=1}^{n} x_j = 1.$$ 

To branch on some $x_j$ by setting it to 0 in one branch and 1 in the other results in an unbalanced tree since the branch with $x_j = 1$ fixes all the other $x$ variables in the GUB constraint to 0, while in the 0 branch the other variables are not affected. Instead a branching can be constructed by adding a constraint

$$\sum_{j=1}^{k} x_j = 1$$

in one branch and a constraint

$$\sum_{j=k+1}^{n} x_j = 1$$

in the other. Note that the constraint $\sum_{j=1}^{k} x_j = 1$ is equivalent to setting $x_j = 0, j = k + 1, \ldots, n$. If $\bar{x}_j$ are the values in the LP-solution then $k$ can be set to $\min\{t : \sum_{j=1}^{t} \bar{x}_j \geq \frac{1}{2}\}$. This constraint branching is a case when branching is not done directly on variables but on other entities (Wolsey [14]).

4.4.5 Active Constraints Branching

A collection of branching methods that all focus on active constraints instead of the objective function was developed by Patel and Chinneck in [10]. The idea is here to find a first integer solution faster than with other branching rules by trying to create two new subproblems that are as different as possible.

The set of active constraints in the current node is the constraints that hold with equality in the LP-solution. All the methods tested in [10] are based on two steps. First each variable is given a weight in each active constraint, and then there is an evaluation based on a combination of all active constraints.

- The weight is based on a combination of two measures. One of these is based on how much the variable can influence an active constraint. Two implemented alternatives of this kind are:

  1. The weight is one if the variable is present in the active constraint and zero otherwise.
2. The weight is \( \frac{1}{\sum_j \|a_{ij}\|} \) (sum the absolute value of all coefficients in the active constraint) if a variable is present and zero otherwise.

The other measure tries to estimate how much a constraint can be influenced. Examples of this are:

1. The weight for a variable that appears in the constraint is one divided by the number of integer variables present in the constraint. If the variable is not present, the weight is zero.

2. The weight for a variable that appears in the constraint is one divided by the number of fractional integer variables present in the constraint. If the variable is not present, the weight is zero.

• Some different ways of combining the two weights were implemented. Two of the given alternatives are:

1. For each variable a score is calculated based on the highest weight in an active constraint and the variable with the highest score is chosen.

2. For each variable the weights in all active constraints are summed and the variable with the highest sum is chosen.
Chapter 5

Methods

In this chapter the methods that have been implemented in this thesis project will be presented. These are: a new problem specific method called “Implication branching”, reliability branching (introduced in 4.4.3), and the method distribution node selection from 4.3.3. In all of the implemented methods information is provided to XPRESS by a type of functions called “callbacks”, which will be explained in section 5.1.

5.1 The XPRESS Solver

XPRESS is a software that solves MIP-problems using branch and bound. It is built to be able to handle all steps in the solution process. It is however also possible for the user to provide information, such as setting new bounds for a variable and/or change the decisions made during the search. This is done with callbacks via an interface written in C++. One of the callbacks that is used to change the decision of XPRESS is called every time a branching decision has to be made. Now the user has the option of deciding what variable or entity to branch upon and/or which branch to investigate first. This callback is used when implementing implication branching (see section 5.2) and reliability branching (see section 5.3). Several callbacks are also used in the implementation of distribution node selection, to handle the set of active nodes \( N \) and to select a node among these (see section 5.4).

Before XPRESS starts to solve the problem there is a presolve step where the problem is modified. At this step some of the variables are removed, variable bounds are tightened, and cuts are added, and XPRESS also performs a heuristic search to find a better starting solution. This initial step is used with all of the methods and for all of the test problems.

5.2 Implication Branching

One idea for finding a good branching rule is to use knowledge about structure that is specific to the particular problem. The method implication branching tries to estimate how important variables are, based on the constraints, and in turn how many variables that can be influenced by a decision. From constraints
3.1, 3.10, 3.11 and 3.3 certain conclusions can be made about the effects of setting a variable to zero or one.

With constraints of type 3.10,
\[
\nu_{\text{pref}(c_1,\bar{s}),c_1} - \sum_{s \in R^+(c_1,c_2,s)} x_{s,c_2} \geq 0,
\]
setting \(\nu_{\text{pref}(c_1,\bar{s}),c_1}\) to 0 forces all involved \(x\) variables to be 0 as well. Setting any of the \(x\) variable to 1 implies that \(\nu_{\text{pref}(c_1,\bar{s}),c_1}\) must be 1.

From seniority constraints of type 3.11,
\[
\nu_{\text{pref}(c_1,\bar{s}),c_1} - \nu_{\text{pref}(c_2,\bar{s}),c_2} \geq 0,
\]
similar conclusions can be made. Setting \(\nu_{\text{pref}(c_1,\bar{s}),c_1}\) to 0 implies that \(\nu_{\text{pref}(c_2,\bar{s}),c_2}\) is 0, and the other way around, setting \(\nu_{\text{pref}(c_2,\bar{s}),c_2}\) to 1 forces \(\nu_{\text{pref}(c_1,\bar{s}),c_1}\) to be 1.

In constraints 3.1,
\[
\sum_{s \in S^2(c)} x_{s,c} = 1,
\]
there is the trivial implication that, setting an \(x\) variable to 1 forces all other \(x\) variables to be 0.

Finally constraints 3.3,
\[
\nu_{v,c} - \sum_{s \in S^2(c):\text{pref}(c,s)=v} x_{s,c} - \nu_{v-,c} = 0,
\]
establish connections between preference and roster variables. Setting \(\nu_{v,c}\) to 0 implies that all variables with a negative coefficient must be set to 0. Setting a variable with a negative coefficient to 1 forces \(\nu_{v,c}\) to be 1 and also the other variables with negative coefficients to be 0.

Also note that the preference variables can be expressed as a sum of roster variables.
\[
\nu_{v,c} = \sum_{s \in S^2(c):\text{pref}(c,s) \leq v} x_{s,c}
\]
Thus branching on a preference variable is equivalent to GUB branching (described in section 4.4.4) on roster-variables. This is because of constraints 3.1, stating that every pilot should be assigned a slot.

5.2.1 Implication Graph

Using the information about implications from the constraints outlined in the former section a graph can be constructed. Two nodes are created for each preference and roster variable, one for the case where the variable is set to one, and one for the case of setting the variable to zero. An arc from node \((i,\alpha)\) to node \((j,\beta)\) is then added if setting the variable \(i\) to the binary value \(\alpha\) implies that variable \(j\) must take the binary value \(\beta\). From this implication graph, information can be found about all preference and roster variables affected by a branching decision on a specific preference or roster variable. In the implementation a list of all implications is calculated and saved for each node.
5.2. Implication Branching

5.2.2 Example

To give an idea of how variables are connected an example of a simplified case will be given. Assume that all pilots have given the same preference order to all slots with \( s_1 \) being the most preferred and \( s_{2m} \) being the least preferred, and that each given preference value is unique (each slot has a different preference value). Also assume that the same seniority order between pilots holds for all slots with pilot \( c_1 \) being the most senior and pilot \( c_{2n} \) being the least senior. Now implications that are made from fixing variable \( \nu_{\text{pref}}(c_n, s_m), c_n \) can be illustrated. The number in the box is the value of the variable that is set and the other numbers are implied fixed variables.

- First is the case where \( \nu_{\text{pref}}(c_n, s_m), c_n \) is set to 1. The matrix to the left contains all preference variables that become fixed in this branch while the matrix to the right contains all roster variables that become fixed. The box in the right matrix corresponds to the position where the preference variable is fixed. It is empty since this fixing has no direct implication on the variable \( x_{s_m, c_n} \).

![Figure 5.1: Preference variables \( \nu \) to the left and of roster variables \( x \) to the right.](image)

The decision to set a preference variable \( \nu_{c, c} \) to 1 can be stated as: pilot \( c \) gets a slot that he gives a value of at most \( v \). This implies that all preference variables that correspond to slots with higher preference values (less preferred) are also set to one. Thus also the roster variables corresponding to higher preference values slots are set to zero. These conclusions are made from the constraints that define preferences (see 3.3).

Seniority rules (3.10 and 3.11) are the reason why preference variables vertically from \( \nu_{\text{pref}}(c_n, s_m), c_n \) are determined. All pilots that are more senior than \( c_n \) must get slots that they value at least as much as \( s_m \).

When \( \nu_{\text{pref}}(c_i, s_m), c_i \) is set to 1 for all \( i \leq n \), the preference constraints can then be used to fix the preference and roster variables in the upper right corner.

- Similarly the matrices in Figure 5.2 show the effect of setting \( \nu_{\text{pref}}(c_n, s_m), c_n \) to 0. To the left are the affected preference variables and to the right are the affected roster variables.
In this case the pilot did not get a slot that he/she values as equal or better than \( s_m \). This directly implies that all preference and roster variables corresponding to slots that are preferred over \( s_m \) are also set to 0.

Now seniority rules \((3.10 \text{ and } 3.11)\) affect all pilots that are less senior than \( c_n \), with the direct effect that none of them can get the slot \( s_m \) or any other slot that is preferred to slot \( s_m \). This also causes the corresponding preference variables to be set to 0. Thus both the preference and roster variable in the lower left corner is set to 0 in the branch \( \nu_{\text{pref}}(c_n,s_m) = 0 \).

To summarize, in each of the branches about one fourth of all preference and roster variables are determined, which is a big reduction in problem size. Since problem size reduction is an important criterion when choosing which variable to branch upon it seems reasonable to use a method that is based on the implication graph. In practice the effect is smaller than this ideal example shows, but the example still illustrates the overall impact of a branching decision.

### 5.2.3 Parameters

In the implementation there are several settings that have to be made. The method has been tested with different parameters that will be explained in this section.

When deciding what variable to branch on, the LP-solution in the current node is available and can be used in combination with the implication graph to make the branching decision. For each choice of setting a fractional preference or roster variable to 0 or 1, the effects on other variables are evaluated. In this thesis the measures used can be divided into four groups:

- Using the LP-solution the total reduced fractionality that results from a fixing can be calculated and used as a measure.

- The number of fractional variables that are affected by a branching decision.

- Without considering the LP-solution, count the number of variables that are fixed as a result of a branching decision (only variables that are not fixated contributes to this measure).
Various combinations such as a weighted sum of these measures were also tested.

The objective of the first two variants is to find a branching variable that has the largest impact on the LP-solution. The idea behind the third one is rather to reduce the size of the LP-problem that has to be solved in every descending node.

For each fractional variable that is a candidate to branch on there are two branches and a combined score is calculated. Two different implementations has been made based on score functions 4.1 and 4.2. Here the notation for the score in the down branch is $S_D$ and in the up branch $S_U$.

- $\alpha_1 \min\{S_D, S_U\} + \alpha_2 \max\{S_D, S_U\}$ with $\alpha_1 = 5/6$ and $\alpha_2 = 1/6$
- $\max\{S_D, \epsilon\} \cdot \max\{S_U, \epsilon\}$ with $\epsilon = 10^{-4}$

The variable to branch on is chosen as the one with the highest score. The reason for these score functions is to make a choice with a large impact on the LP-solution in the current node, but also to create a branch and bound tree that is balanced.

After a variable has been selected it is also necessary to decide which branch to investigate first. The following options have been tested.

- Up branch.
- Down branch.
- Keep the decision from XPRESS.
- Choose the branch with the maximal score ($\max\{S_D, S_U\}$).

Some of the integer variables (the two most important are course variables and variables concerning the option to hire one or more pilots) can not be included in the implications used for implication branching. The idea is to let XPRESS decide which variable to branch on in some parts of the branch and bound tree, and a parameter was used to control where.

- Depth-based control. Use implication branching to choose a variable to branch on for all nodes above a certain depth of the branch and bound tree. In nodes below this depth XPRESS decides branching variable.
- Value-based control. Calculate the score of the best branching variable according to implication branching and if this value is above a threshold, branch on that variable. Otherwise let XPRESS decide which variable to branch on.

5.3 Reliability Branching

Pseudo-costs are calculated as averages over samples gained from strong branchings. In each node there is an initial step where strong branching is performed on all candidate variables with uninitialized pseudo-costs. As in implication branching, a method to combine pseudo-costs in the up and down branch has to be designed. The same two rules as in implication branching has been tested.
in reliability branching. Also when deciding which branch to select first the same rules as in implication branching has been tested. In each case where strong branching is performed the maximal number of dual simplex iterations that are allowed to make should be supplied. Several different values, depending on problem size, were tested. Based on results by Achtzberg et al. in [2], each pseudo-cost was updated at most four times in total. When searching for a branching variable, scores for all candidate variables are sorted in descending order and pseudo-costs are updated (if they have been updated less than four times) starting with the variable with best score. If no new highest score has been found in four consecutive updates, the variable with highest score is chosen.

In the cases where the strong branching iterations arrived at the conclusion that the problem is infeasible (or that the value was worse than the best found integer solution) an estimate of the pseudo-cost of the variable is required. Here a few different options was tried.

- Use the value returned by XPRESS from the callback that performs strong branching, which is usually very large.
- Estimate the pseudo-cost based on the value of the best found integer solution.
- Do not consider the variable when branching in this node.

Since pseudo-costs from such cases tended to be much larger than other pseudo-costs, tests were performed where pseudo-costs was updated only locally. Thus the effect was seen only in the current node.

### 5.4 Distribution Node Selection

The implementation was done based on the article [13] by Wojtaszek and Chinneck. Each time XPRESS asks for a node to be selected the mean value and standard deviation of the LP-values in the nodes investigated so far are calculated. These values are also calculated for fractionality, but every pilot (considering both preference and roster variables) contributes with at most one to the number of fractional variables if any roster or preference variable is fractional. Since the children of a node initially inherit the values (LP-value and number of fractional variables) of their parent, which child to choose must be possible to decide in cases where both children are uninvestigated. Based on results by Achtzberg in [1], the node that is a result of the up branch is selected first.

An important parameter to set is when distribution node selection should be used. Two alternatives has been investigated.

- Only use this method to select a new node when XPRESS chooses to backtrack (a selection of a node that is not a child of the latest investigated node).
- Use this method to choose a new node among all active nodes every time a node has been investigated. This is generally more time-consuming but might be worthwhile if it leads to fewer unnecessary nodes being investigated.
5.5 Problem Reductions

Based on a heuristically acquired integer solution some of the rosters in the original problem are removed. Consider a roster variable \( x_{s,c} \) that is 0. For \( x_{s,c} = 1 \) to be feasible, all pilots that are more senior than \( c \) with respect to \( s \) must get slots that they are at least as content with as with slot \( s \), and no pilot that is less senior than \( c \) can get a slot that \( c \) would prefer to slot \( s \). How many of the other pilots that have to change slot, if \( x_{s,c} = 1 \) (in the current integer solution) can be calculated. If it is requires that more than a certain number of pilots change slots then the roster variable \( x_{s,c} \) is removed from the problem. This process is repeated for all roster variables that are zero.

It might be the case that an optimal solution is cut away in this step, but the reduction in problem size is considered to be more useful in practice since it is quite likely that the removed roster variables will be zero in good integer solutions.

This reduction also affects the size of the implication graph. Typically the kept roster variables looks like Figure 5.3. Symbols “X” at the lower left and upper right corners indicate that variables have been removed. However the exact number of kept variables depends on the problem instance and integer solution. With part of the variables removed, the average number of implications for each preference and roster variable is lower than indicated in the simplified example 5.2.2.

5.5.1 Redundancy

Consider a slot \( s \) with three pilots that have the seniority relations \( \text{sen}(c_1, s) < \text{sen}(c_2, s) < \text{sen}(c_3, s) \), and where the following three seniority constraints are thus included in the problem formulation.

\[
\nu_{\text{pref}(c_1, s), c_1} - \nu_{\text{pref}(c_2, s), c_2} \geq 0 \\
\nu_{\text{pref}(c_2, s), c_2} - \nu_{\text{pref}(c_3, s), c_3} \geq 0 \\
\nu_{\text{pref}(c_1, s), c_1} - \nu_{\text{pref}(c_3, s), c_3} \geq 0
\]

Figure 5.3: Removed roster variables.
If the first two constraints are satisfied then the third constraint also holds and is therefore redundant. This allows a part of the seniority constraints in 3.2 to be removed from the model. Before a problem is set up as a mixed integer program there is a check for redundancy among the seniority constraints.
Chapter 6

Results

6.1 Test Problems

The developed methods have been tested on seven problems from four airlines. In six of these seven problems a feasible starting solution is given. For each of the seven problems, 15 instances have been created by randomizing the order of a subset of the constraints. This is done because the solutions sequence varies significantly between equivalent problem instances where the order of the constraint matrix has been changed.

For all of these airlines the pilots bids for positions and corresponding preference values are modeled according to the rules at the specific airline.

The problems originate from real airlines but the airlines will be referred to as Red, Blue, Black and Green in this thesis due to confidentiality. Aircrafts Antonov An-124 and Embraer ERJ 135 ER that are used in the examples below are not used at any of these airlines.

6.1.1 Airline Blue

The methods were tested on two instances related to airline Blue, Blue-small and Blue-large. In both cases the planning problem is for one year with two bidding periods. It is always considered to be better to get a promotion in the first bidding period than in the second. The main property that differs between the two instances is how detailed the seniority description is. In the smaller problem it does not matter when in a bidding period that the promotion occurs, while in the more detailed case it is always better to get a promotion earlier. This has the effect that the detailed problem has a lot more preference variables because there are more different preference values. This effect can be seen when comparing Tables 6.1 and 6.2.

There is no limitation on used simulators since they can be leased when needed. Thus the resource constraints 3.5 are not present for simulators.

Both of these problems are possible to solve to optimality within one hour.

Examples

A promotion to Antonov An-124 is preferred over a promotion to Embraer ERJ 135 ER. How the preference values for promotions in different weeks are modeled
in problem Blue-small are presented in the following table. Here, each week in a period has the same preference value.

<table>
<thead>
<tr>
<th></th>
<th>Period 1</th>
<th></th>
<th>Period 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W 1</td>
<td>W 2</td>
<td>...</td>
</tr>
<tr>
<td>Antonov An-124</td>
<td>1</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>Ambras ERJ 135 ER</td>
<td>2</td>
<td>2</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 6.1: Blue-small: Weekly preference values for two possible promotions.

The next table shows how the preference values are modeled in the more detailed problem. A promotion to Antonov An-124 is again preferred over a promotion to Ambras ERJ 135 ER.

<table>
<thead>
<tr>
<th></th>
<th>Period 1</th>
<th></th>
<th>Period 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>W 1</td>
<td>W 2</td>
<td>...</td>
</tr>
<tr>
<td>Antonov An-124</td>
<td>1</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>Ambras ERJ 135 ER</td>
<td>27</td>
<td>28</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 6.2: Blue-large: Weekly preference values for two possible promotions.

6.1.2 Airline Green

The planning problem is for four years, with two bidding periods in the first year and none in the last three years. The reason to have extra time is to make sure that enough pilots are trained also in a long-term perspective. Promotions in the first period are locked so it is only decisions in the second bidding period that have to be made.

When comparing promotions, bidding period is the most important factor and an earlier promotion is preferred. In each bidding period position is more important than time. Thus a pilot prefers to wait for a promotion if the later promotion leads to a position that is considered better. Time within a bidding period is only considered when comparing two pilots that have the same position. Therefore the seniority constraints look a bit different depending on if two pilots have the same position or not. To some extent it is possible to move work hours between time periods, giving some extra flexibility.

This problem has not been possible to solve to optimality (within a reasonable time limit).

6.1.3 Airline Black

The setup and preference system in airline Black is similar to the one used in problem Blue-large. The problem has not been solved to optimality.

6.1.4 Airline Red

Due to confidentiality, no description of the seniority model used at airline Red will be given.
Three problems, called Red-small, Red-medium and Red-large are modeled with the main difference in the amount of details. The large and medium problems have 13 time periods while the smaller problem has only three time periods. The problem Red-medium is obtained from Red-large through large reductions in demand for all positions. This results typically in a solution where fewer pilots are promoted. All three problems are simplified versions of the original one.

The three problems vary significantly in how difficult they are but none of them are possible to solve to optimality within a reasonable time limit.

Parameters were tuned only for problems Red-small and Red-large. For problem Red-medium parameter values from the other two problems were used.

6.2 Performance Measures

When comparing methods there are two things that need to be decided. The first is which value or property of the solution to compare. The second part is how to compare them.

In this thesis project the object of primary interest is the value of the best found integer solution. In the harder problems a time limit was set and then the best integer solution that had been found at this time was used for comparison. These time limits were based on an estimate of how long time it would take to investigate 100,000 nodes.

For the two problems where an optimal solution was expected to be found the value for comparison was instead time needed to verify optimality. For the 15 perturbations of the problems that were solved the arithmetic mean value was calculated as well as the standard deviation. For both these numbers, a smaller value indicate a better method. The implemented methods were compared with XPRESS, using standard settings (with the addition to solve the initial LP-problem with the barrier method, which all of the methods used).

6.3 Test Setup

The number of possible parameter values and variations of methods was considered too large to test them all. The methods were thus initially tested on five perturbations (where the order of all the constraints was randomized) of problem instances Red-small, Red-large, Green, Blue-small, and Blue-large to get an indication of how the methods performed on each instance. For each problem, those with the best results from this preliminary step were then tested on a set of 15 perturbations of the problem where only the order of the constraints that force each pilot to get a single roster were randomized. The reason for this was that the preprocessor in XPRESS prefers similar constraints to be placed next to each other. These results were used to evaluate the methods. Methods based on reliability branching seemed to perform worse than methods based on implication branching, node selection, and combinations of these two, and thus this method was not tested as thorough as the other methods.

The two problems Red-medium and Black was then used to test how the methods performed on problems when no tuning was done.
The costs in the problems are not possible to directly translate to monetary units since there are soft costs included. Some costs that are not affected by the planning are removed when the instance is set up. Thus it is not possible to compare costs for different problems straight ahead.

6.4 Test Results

The following methods were tested on at least one problem. Abbreviations are BI for implication branching, BP for reliability branching and NS for node selection.

Default XPRESS used without any callbacks.

BI-up A method based on implication branching. Only the amount of reduced fractionality was taken into account. Branching direction was up and the additive score function was used. Implication branching was used on all nodes on depth smaller than or equal to 20.

BI-most A method based on implication branching. Same settings as in BI-up were used with the only difference in branching direction. In this method the direction with the largest score was chosen.

NS-def A node selection method that backtracks only when XPRESS choose to backtrack. Branching direction was up.

NS-back A method based on node selection but with backtracking every time a new node is chosen. Branching direction was up.

BP-1000 A method based on reliability branching. In cases where results from strong branching were replaced with an estimate, this was made only locally (in the current node) with the value of the best found integer solution. Branching direction was up, the score function used was the multiplicative, and the maximal number of dual simplex iterations performed with strong branching was 1000. This number is about twice the average number of dual simplex iterations performed in each node (for airline Green).

NSBI-up Implication branching used together with node selection. The settings in the implication branching was the same as in BI-up and the settings in the node selection was the same as in NS-def.

In the following sections, results for a selection of these methods will be presented for each problem.
6.4. Test Results

6.4.1 Red-small

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean Value</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>23,767,466</td>
<td>44,214</td>
</tr>
<tr>
<td>BI-up</td>
<td>23,733,839</td>
<td>40,591</td>
</tr>
<tr>
<td>NS-def</td>
<td>23,754,831</td>
<td>63,216</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Mean Value</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS-back</td>
<td>23,768,771</td>
<td>57,153</td>
</tr>
<tr>
<td>NSBI-up</td>
<td>23,691,317</td>
<td>21,450</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison of objective function values after 5000 seconds.

As Table 6.3 shows, two methods perform better, giving lower objective function value and standard deviation, compared to the Default method.

In Figure 6.1 the method Default is compared with the methods BI-up and NSBI-up, with intermediate objective function values shown for every 500 seconds up to 15,000 seconds.

![Figure 6.1: Results for problem Red-small. Progress in mean objective function value over time. The lower bar in the graph corresponds to the best found integer solution.](image)

This graph clearly shows that the two implemented methods perform better than method Default at all times. Based on this graph, the time it takes until the mean value is below a certain value can be used to compare methods (see Table 6.4). Best and worst solution for the 15 perturbations are listed in Table 6.5.
Chapter 6. Results

<table>
<thead>
<tr>
<th>Mean value</th>
<th>Default</th>
<th>BI-up</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>23 800 000</td>
<td>3000</td>
<td>2500</td>
<td>2000</td>
</tr>
<tr>
<td>23 710 000</td>
<td>15 000</td>
<td>7000</td>
<td>3500</td>
</tr>
</tbody>
</table>

Table 6.4: Time in seconds for when the mean value is below the value in the column to left. Data points are at every 500 seconds and the number in the table is the time of the first data point when the mean value is below the threshold.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>BI-up</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best solution</td>
<td>23 695 657</td>
<td>23 679 523</td>
<td>23 649 026</td>
</tr>
<tr>
<td>Worst solution</td>
<td>23 848 385</td>
<td>23 817 340</td>
<td>23 734 869</td>
</tr>
</tbody>
</table>

Table 6.5: Best and worst objective function value after 5000 seconds.

6.4.2 Red-large

<table>
<thead>
<tr>
<th>Mean value</th>
<th>Default</th>
<th>BI-up</th>
<th>NS-def</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 305 739</td>
<td>16 530 490</td>
<td>16 448 671</td>
<td></td>
</tr>
<tr>
<td>Std dev</td>
<td>158 082</td>
<td>224 917</td>
<td>191 217</td>
</tr>
<tr>
<td>Mean value</td>
<td>16 524 383</td>
<td>16 373 968</td>
<td></td>
</tr>
<tr>
<td>Std dev</td>
<td>219 384</td>
<td>189 571</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.6: Comparison of objective function values after 72 000 seconds.

From the results in Table 6.6 can be seen that the method NSBI-up seems to perform better than method Default due to a smaller mean objective value, even though the standard deviation is higher. Also when comparing the best and worst integer solutions among the 15 instances, the method NSBI-up performs better.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best solution</td>
<td>16 276 362</td>
<td>16 025 300</td>
</tr>
<tr>
<td>Worst solution</td>
<td>16 777 178</td>
<td>16 725 657</td>
</tr>
</tbody>
</table>

Table 6.7: Best and worst objective function value after 72 000 seconds.
6.4. Test Results

To better illustrate how the solutions of the different methods improve, the following graph shows a comparison between the method Default and the method NSBI-up with results every 500 seconds up to 144,000 seconds.

![Graph showing comparison between Default and NSBI-up methods](image)

**Figure 6.2:** Results for problem Red-large. Progress in mean objective function value over time. The lower bar in the graph corresponds to the best found integer solution.

As can be seen, the method NSBI-up gives a lower mean objective value at all data points. The difference is quite large in the first half of the search but diminishes when the curve for the method NSDI-up is flattening out. The following table shows a few examples of solution times where the mean objective value is below a certain threshold value.

<table>
<thead>
<tr>
<th>Mean value</th>
<th>Default</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>17,000,000</td>
<td>27,000</td>
<td>12,500</td>
</tr>
<tr>
<td>16,700,000</td>
<td>48,000</td>
<td>17,500</td>
</tr>
<tr>
<td>16,400,000</td>
<td>95,500</td>
<td>47,500</td>
</tr>
<tr>
<td>16,350,000</td>
<td>114,500</td>
<td>95,500</td>
</tr>
</tbody>
</table>

**Table 6.8:** Time in seconds for when the mean value is below the value in the column to left. Data points are at every 500 seconds and the number in the table is the time of the first data point when the mean value is below the threshold.
6.4.3 Red-medium

Methods Default and NSBI-up were tried on problem Red-medium with a time limit of 36,000 seconds. Results are shown in Table 6.9 and Figure 6.3.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>7,078,392</td>
<td>7,036,168</td>
</tr>
<tr>
<td>Std dev</td>
<td>47,871</td>
<td>49,089</td>
</tr>
</tbody>
</table>

Table 6.9: Comparison of objective function values after 36,000 seconds.

From the presented data it is clear that the method NSBI-up performs well compared to the method Default.

6.4.4 Blue

For these problems a limit of 100,000 nodes was used. In the cases where the optimal solution was not proven (or even found) the time to search through 100,000 nodes was used as result. For problem Blue-small the number of problem instances solved to optimality is given in the table. For Blue-large all methods succeeded in solving all 15 problem instances to optimality and thus only solution time is given.
6.4. Test Results

Blue-small

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>BI-up</th>
<th>NS-def</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value</td>
<td>524</td>
<td>1543</td>
<td>802</td>
<td>1040</td>
</tr>
<tr>
<td>Std dev</td>
<td>717</td>
<td>2153</td>
<td>900</td>
<td>1243</td>
</tr>
<tr>
<td>Solved instances</td>
<td>14</td>
<td>11</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 6.10: Seconds needed to find an optimal solution.

Blue-large

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>BI-up</th>
<th>NS-def</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value</td>
<td>4751</td>
<td>3225</td>
<td>3486</td>
<td>2973</td>
</tr>
<tr>
<td>Std dev</td>
<td>1829</td>
<td>773</td>
<td>842</td>
<td>557</td>
</tr>
</tbody>
</table>

Table 6.11: Seconds needed to find an optimal solution.

There is a large difference in performance of method Default compared to method BI-up for problems Blue-small and Blue-large. The results show clearly that the implemented methods performs well on problem Blue-large but worse than the method Default on problem Blue-small.

6.4.5 Green

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>BI-most</th>
<th>NS-def</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean value</td>
<td>-9440.307</td>
<td>-9334.177</td>
<td>-9056.020</td>
</tr>
<tr>
<td>Std dev</td>
<td>231.264</td>
<td>428.732</td>
<td>390.442</td>
</tr>
<tr>
<td>Mean value</td>
<td>-9201.910</td>
<td>-9313.685</td>
<td>-9204.459</td>
</tr>
<tr>
<td>Std dev</td>
<td>698.111</td>
<td>252.810</td>
<td>282.425</td>
</tr>
</tbody>
</table>

Table 6.12: Mean objective function values after 36 000 seconds.

With the lowest mean value and smallest standard deviation it is clear that the Default method performs better than the implemented methods on problem Green. In Figure 6.4 the method Default is compared with the method BI-most with results recorded every 500 seconds up to 72 000 seconds.

Even though there are a few data points where method BI-most shows a smaller mean value, the method Default performs better at most data points. The same conclusion holds for the standard deviation, which is generally lower in the method Default.

Among the 15 perturbations there is one instance that the method BI-most handles worse than the others. When comparing the worst and the next worst solution value this is quite clear. Since stability in performance is regarded as being important, this is a drawback of the method BI-most.
Figure 6.4: Results for airline Green. Progress in mean objective value over time. The lower bar in the graph corresponds to the best found integer solution.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>BI-most</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst solution</td>
<td>-9 280 241</td>
<td>-9 042 818</td>
</tr>
<tr>
<td>Next worst solution</td>
<td>-9 373 022</td>
<td>-9 344 063</td>
</tr>
</tbody>
</table>

Table 6.13: Worst and next worst objective function value after 72 000 seconds.
6.4.6 Black

The test setup for problem Black was similar to the one for problem Red-medium. The methods Default and NSBI-up were compared with a time limit of 36 000 seconds.

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>NSBI-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>105 251 347</td>
<td>106 634 293</td>
</tr>
<tr>
<td>Std dev</td>
<td>600 025</td>
<td>1 198 403</td>
</tr>
</tbody>
</table>

Table 6.14: Comparison of objective values after 36 000 seconds.

Based on the results in Table 6.14 and Figure 6.5, the method Default is here a clear winner.
Chapter 7

Discussion

The purpose of this thesis project was to improve the ability to solve MIPs arising in the airline industry by reducing solution times, and to be able to solve larger such MIPs. Since MIPs are solved at several steps in the planning process (as described in Chapter 1), a small improvement in performance for each MIP involved might add up to a large improvement in the end. In the results for airline Red the solution time is shortened significantly, while keeping the same level of performance with respect to objective value. It was also seen in all versions of airline Red that implication branching was able to handle some variations in problem data when using the same seniority rules and still produce good results. Thus there is a good chance that an improvement in performance could be seen when the method is integrated with the current planning process.

An aspect of shortening solution times is that it might then be possible to test more scenarios. Evaluations of more scenarios can be used to produce a solution that is robust with respect to variations in problem data. It could also support the airline to evaluate different seniority rules or other possible rules such as pay-protection.

One hypothesis was that methods based on implication branching should perform better when preferences are more detailed. This was tested in the two problems, Blue-small and Blue-large, with results that support the hypothesis. One explanation for this comes from the fact that there is a lot more preference variables in Blue-large (1921, compared to 259 in Blue-small). Thus there is more information available that can be used when a branching variable is decided.

The standard deviation in solution times for problem Blue-small is about as high as the mean value of the time when the optimal solution was proven. This indicates that the variation is higher for this problem and that conclusions are less certain compared to the other problems. One possible cause for the variation is the existence of many alternative solutions (in objective function value), since all slots that lead to the same position in the same period have equal preference values, and hence can be switched between pilots while keeping the same objective function value. For Blue-small all methods show tendencies of getting stuck at near-optimal solutions.

An observation from the results is that even though the same mathematical model can be used for all airlines, there is enough difference between the problems of the airlines to cause variations in the results for different methods.
Results from airline Black indicate that it is necessary to tune parameters of the methods separately for each airline to produce good results.

7.1 Future Work

By using optimization approaches to choose values of the parameters within the methods there will probably be a significant improvement in the results. Such a process might also give a better understanding of which parameters that are crucial for the performance of each method.

There is also a lot of parameters involved in XPRESS and tuning these parameters might be well worthwhile. One idea for improvement concerns the presolve step. It is unlikely that it will be beneficial to remove it completely but a modification might give an improvement. In the presolve step a large portion of the preference variables is removed. Keeping some or all of the preference variables could give more efficient methods based on implication branching. This tuning is sensitive though since the presolve step has a large effect on the overall performance.

An attempt to be able to draw more reliable conclusions was to solve 15 perturbations of each problem, but it is still important to be careful when analyzing the results. Both additional problems from other airlines and more test problems for each airline should give results that are more reliable.

For further testing and accurate evaluation it is also important to incorporate the methods in the whole planning process. Results from this thesis indicate that performance for some problems can be improved significantly by using the method NSBI-up.
Bibliography


Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a period of 25 years from the date of publication barring exceptional circumstances. The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtidiga ersättare - under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår. Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och får undervisning. Överföringen av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. Får att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art. Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsmann i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant samband som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

© 2012, Björn Morén