Using a More Fine-Grained Cost-Function for Crew Rostering at SAS

Anders Musikka

Examiner
Dr.-Ing. Christian Schulte

Master of Science Thesis
Stockholm, Sweden 2006

ICT/ECS-2006-17

Abstract

This master thesis examines the merits of using a more fine grained objective function for automatic rostering of airline crew at SAS.

Creating efficient schedules for crew members at a modern airline is a complex and difficult task. A large set of rules (government laws and union agreements) apply, regarding work hours, rests, training etc. Computer support is used to create efficient schedules. A cost function is specified. The computer system tries to minimize the value of the cost function while not breaking any rules.

The hypothesis has been made, that some of the rules used are too rigid. That is, some of the rules used within the system disallow some schedules which are in fact desirable.

Some of the rules within the system are government laws. These may be course-grained, but can, nonetheless never be broken.

This work studies some rules which may be reimplemented in a more fine-grained fashion within the SAS scheduling optimization environment. Metrics are introduced, and results evaluated.

Two rules are reworked, one only slightly, and the other extensively. A case is made, for the first one, that a small amount of money is saved. A case is made, for the second one, that it allows some of the worst schedules, which would otherwise be created, to be avoided.

Contents

Chapter 1

Introduction

Crew Scheduling, in the airline industry, is the act of determining which flight each individual cabin crew member is to work on. There are many hard constraints, such as not breaking laws and union agreements. It is also desirable to make the schedule reasonably insensitive to delays (to have margins), and to maximize "Quality of Life", for crew.

At SAS, this problem is solved using software from Gothemburg-based Carmen Systems AB. The problem is divided into two parts, Pairing and Rostering, which are described in 3. This thesis only concerns Rostering.

In this thesis, two modifications to the crew scheduling model used at SAS are studied:

Minimum Rest Time A more fine grained model of the Minium Rest Time-constraint is developed (see 8.1).

"TP" A model that estimates the work load of shifts (see 8.2).

Apart from describing these two modifications, this report also strives to serve as a self-contained, if very basic, introduction to the field of crew scheduling, deliberately spending many pages to describe the real world airline operations that the scheduling is set to serve. It is the intention that a person acquainted with neither Operations Research nor the airline industry, should be able to read and understand this thesis.

A recipe for modeling undesirable schedule properties is developed (see 7.7), and then tried on the "Minium Rest Time"-constraint.

Many schedules have been created, and the results are presented in 9. For the Minimum Rest Time-constraint, a modest improvement in schedule quality is demonstrated. The TP system is shown to produce reasonable schedules, but if there is an increase in the "Quality of Life" of crew, has not been determined.

Chapter 2

Airline Operations

Since people want to go flying any day of the week, not just weekdays, and early mornings as well as late evenings, it is obvious that the pilot or airhost/airhostess profession can not be a 9-to-5 kind of job. Instead, crew typically work a few days in succession, then have a few days off, then work a few days, etc. A sequence of uninterrupted work days is called a ’working period’ (sometimes referred to as a WOP). Working periods are between 1 and 5 days long. Nights are often spent at hotels away from base. Each day is often referred to as one ’duty’. A take off, and the period up to and including the following landing, is called one ’leg’. Each duty usually consists of several legs (about 2 to 6). In this work, the word ’flight’ is sometimes used for leg. In this text, these two words are equivalent.

A distinction is made between short-haul and long-haul flights. There is no universally acknowledged exact definition. For example, at SAS, all flights within Europe are short haul, and flights to the US or east Asia are long haul. Long haul flights usually involve staying at the destination for at least one night, before flying back. For short haul, the crew typically flies back with the plane they arrived in

2.1 Block time

Wheel-blocks are put in front of the aircraft’s wheels whenever it is on ground, to prevent any unwanted motion due to gusts or the like. Just before the aircraft is to depart, the wheel blocks are removed.

This explains why the period from push-back (departure from the gate) to arrival at the destination airport gate, is called the ’block time’. It is simply the length of time during which the wheel-blocks are not in place. Notice that this time is somewhat longer than the airborne time, and in the case of a congested runway before takeoff, can be significantly longer (many hours). Sometimes you will see the terms block-off for the time when the wheel-blocks are removed, and block-on for when they are put on.

2.2 Standby

The value of the income for a single flight is typically very large. Canceling a flight with 200 paying passengers is likely to cost, at least, in the order of several hundred thousand SEK, and it often takes time to recover from interruptions of this kind. If one flight is canceled, any other flight that was planned to be flown by the same aircraft is likely to be delayed, or canceled as well. When many flights have been canceled (for example due to a violent snow storm at an important airport), it has taken some airlines several days to recover and return to planned operations.

The result of this is that airlines try very hard to avoid canceling flights. As a part of this effort, aircrew are kept on ’standby duty’. This usually means that they can stay home, until someone calls in sick, at which point they have to travel to the airport for active duty. Standby duty is considered unattractive by crew. At first this may seem surprising, since there’s a good chance a person on standby won’t have to work at all. However, people do have a great dislike for uncertainty.

At SAS there is also something called a “blank day”. This is a day on which the crew member could work, according to the agreement, but on which there is no production for him/her (no aircraft to fly). A large number of blank days is indicative of surplus crew employed. A blank day works very much like a day with only standby duty. The biggest difference between a blank day and standby duty is that if a crew member is going to be ordered to work on a blank day, he must be ordered to do so in the afternoon the day before, at the latest [11].

The intention of blank days is often described as being to signal to the Union, to crew, and to management, that there are redundancies in crew. However, over the years, the blank days have become (partially) relied upon as extra standby. The need for ’regular’ standby is decreased, because of the existence of blank days.

2.3 Hub and Spoke

The set of available flights and destinations for an airline can be viewed as a graph, with destinations as nodes, and flights as edges. How this network should look is a subject much studied by airlines.

Naturally, you want to include edges which are very profitable. But you also want to make sure the graph stays connected - there are many reasons for this. One reason is that it allows passengers to get from each node to every other node. Another reason is that most (or all) airlines have come to the conclusion that there are good reasons to have what is called ’bases’. A base is an office area for airline crew, where they receive their schedules, where they can have briefings and debriefings, meetings and different types of ground duties, and where management has its office. The base is also the location from which all working periods start, it is the location where aircrew ’check in’ and ’check out’. It is the formal location of employment for employees. Therefore, at the very least, all nodes should be connected to at least one base.

There are reasons why a few large bases, rather than many small bases, are preferable. For one thing, there’s of course economies of scale. Another reason is something as unexpected as the fact that the "Law of Large Numbers" makes it possible to use less crew on standby, while still not increasing the risk of having to cancel a flight due to man power shortage.

Many airlines believe that only flights starting or landing on a base should be created. One attractive feature of such an organization is that the effects of delays are kept under control. Cancelling one flight is likely to only affect the following flight. This sort of organization is often referred to as "Hub and Spoke".

SAS Sweden only has one base – Stockholm/Arlanda. It is believed that no other city/airport has enough traffic to create a base for SAS of sufficient size. SAS is a Hub

Figure 2.1: An SAS aircraft at work. This is the De Havilland Canada Dash 8 Q400, commonly known as the “Q400”. The Q400 is normally flown with two air hostesses. It has two engines of 5000hp each, composite 6 bladed propellers, a cruise speed of 670km/h, and takes around 70 passengers. Thanks to Hans Norman/www.scanliners.com for the picture. Please visit http://www.scanliners.com/ for pictures of the other aircraft in the SAS fleet!

and Spoke-airline.

2.4 Crew as passengers

Of course the entire point of the airline is to transport paying passengers (or cargo). But aircrew may sometimes themselves fly as passengers on a plane, for instance, in order to get to an airport where an otherwise unmannable flight begins. A flight like this, without active duty, is called a ’deadhead’. Crew on this kind of flight are said to be deadheading (or, in some texts, passengering). Since deadheading crew take up seats, and do not contribute to production, deadheads are undesirable, if sometimes unavoidable. At times, crew has to be sent deadheading with another airline. This is considered very expensive.

2.5 Layovers

Because they are expensive, aircraft are put to work as much as possible each day. Typically the first flight of the day for a short haul aircraft occurs around 06:00, and the last around 23:00. This means that each aircraft is flown by at least two teams of crew members, each day. Sometime during the day, a change of crew has to occur. Since crew changes are most efficiently done at the base, this is yet another advantage of the hub and spoke-model described earlier.

Naturally, each crew member would like to end his/her shift at his/her home base, and they cannot start a working period in any other place than the base. Even in the hub and spoke-model, however, crew sometimes have to have a layover somewhere.

The typical situation when this occurs, is when an early morning, very profitable, flight has been identified. An example: It is determined that the 06:25 flight from Helsinki to Stockholm is a very profitable one. This means that there must be an SAS aircraft physically available in Helsinki at ca 06:00. The plane must arrive at least 20 minute before it is to depart. This is to allow time for passengers to embark, the plane to be fueled, and the pilot to make his external inspection, among other things. Either it flew there the day before, or it flew there in the very early morning. It will be difficult to sell tickets for a 04:00 flight from Stockholm to Helsinki, so the former choice is made. The airplane is likely to fly to Helsinki around 23:00 in the late evening. A problem that now occurs is that the crew that flew there late in the evening, will probably not be able to fly the plane back to Stockholm in the morning, due to government rules for minimum rest time between two working days.

What is typically done in this situation is that a different crew is flown to Helsinki sometime during the previous day. This crew is then used for the early morning flight, and then the late night crew gets to fly home later during the day. This means that two full crews (approximately a dozen people) have to stay at a hotel in Helsinki, as a consequence of the decision to fly the early Helsinki–Stockholm leg.

2.6 The Law

There are many laws that airlines have to abide by. For crew scheduling, the intent of the law is to make sure that airlines do give their Cabin Crew adequate rest, and a reasonable work load, in order for them to be able to ensure flight safety. All Cabin

Crew-members are trained to aid in the evacuation of the airplane, and must be able to do this in case of an emergency.

In Sweden, the most relevant text is “BCL-D 1.15, Tjänstgöringsbegränsningar för besättningsmedlem”. This is a 5 page document which imposes a couple of rules on how much crew members may work. Actually, the document only governs *planned work*, it does not say that the rules must be followed exactly in operation. However, the document also says that the planning must be *realistic* (without describing what is meant by realistic), thereby eliminating any loopholes. Arguably the most important rules in this document are the “90-point rule” and the “270-point rule”. To understand these rules, we must first describe what “points” are (these are defined in the BCL-D 1.15-document).

6 points are awarded for each hour the crew member works (8 points/h for night duty). Also, 5 points are awarded per landing, however, the first two landings are free (without points) for cabin crew. As an example, 10 hours of work, with 4 landings, for a pilot, would be $10 \times 6 + 4 \times 5 = 80p$.

The 90-point rule now says that between any two periods of “night rest” (sleep, that is), a crew member may have at most 90 points of duty. This rule governs how much work a crew member can have in any one day.

The 270-point rule says that in any 168 hour-period, a crew member may have at most 270 points. This rule governs how much work a crew member can have over a week long period. Note that this rule governs duty within a “sliding window”, it is to be evaluated for *every* un-interrupted 168h-period.

Chapter 3

Basics of crew planning at SAS

There are many planning tasks that need to be performed for the operation of an airline. Three of these are timetable creation, pairing and rostering.

3.1 Creating the timetable

Creating the timetable entails deciding the where (between which airports), the when, and the what (what aircraft type) of flights. An example might be: “On the 10th of September, at 10:40AM, Airbus A330 flight SK903 leaves Stockholm Arlanda Airport for New York Newark airport”. For some flights there is also something called the Onward flight reference (OFR). This is an indication of which flight the aircraft will fly after completing the current flight. In other words, the following items are decided:

- Flight Number (e.g. SK903)
- Identity of departure and arrival airport.
- Departure and arrival time.
- Aircraft type
- Which flight number the aircraft will fly after landing.

However, it is not decided:

- Which actual aircraft individual will fly the trip.
- Who the pilot, and co-pilot will be.
- Who the flight attendants will be.

3.2 Pairing

Pairing involves creating anonymous spells of work. A pairing is a sequence of flights that one crew member might fly. It has the property that it is possible for someone with the right qualifications to fly this sequence without violating union agreements, government laws or other constraints. A large number of pairings are created, enough so that exactly all flights are manned. The reasoning behind the word “pairing” is that each pairing is usually to be flown by two (or more) crew members. For example, a pairing might be a sequence of flights that is to be flown by one pilot and one co-pilot working together. A pairing may be one work day, or a couple of work days long.

3.3 Rostering

In this step it is selected which crew-members are to fly each pairing. It is in this step that the actual schedules for crew members are created.

Creating the timetable, pairing, and rostering, must be done in this order. The reason that each step is done separately is purely a means to reduce the complexity of the problem, to make it more tractable. There is a definite risk of sub-optimization – a risk that can be alleviated slightly through feedback to previous steps, with iteration. That is, first the time table is created, then pairing is done for this time table. If, in the pairing step, it is discovered, that a specific flight is difficult to crew efficiently, the time table may be changed, and pairing done again.

The most basic property a schedule must have, is that it must be physically feasible. Obviously infeasible schedules are those that require a crew member to fly two aircraft at the same time, or require him/her to move from one airport to another without being provided any means of transportation. Also, ideally, all legs should be active legs (legs which are not deadheads).

This paper focuses on Rostering, with some mentioning of Pairing. At SAS, these two activities are performed at the same department, Crew Planning, whereas timetabling is done elsewhere.

SAS uses software from Gothenburg-based Carmen Systems AB.

3.4 The life-cycle of a plan

The crew planning at SAS is done in month-sized chunks. Let us take September 2005 as an example. Somewhere in July, the process starts. By this time, it is expected that the September timetable has been created. The pairing problem is solved, followed by the rostering. In the middle of August, the finished plan is published, and the individual crew members get to know when and where they will be going in September. Now the plan is handed over to Roster Maintenance. One thing is sure, disruptions to the plan will occur. Example causes are illness of crew, inoperable aircraft, and license issues . These issues are handled by Roster Maintenance, by using standby, blank days, or by asking crew to work in their spare time (with greatly increased salary). At the present time, roster maintenance requires a lot of manual intervention. An overview of crew scheduling in general is given by Medard et al [4].

Chapter 4

Modeling Cost

The biggest costs for an airline are (in no particular order):

- Investment cost of aircraft.
- Fuel costs.
- Salaries.

Aircraft are tremendously expensive. The cost of an Airbus A340 (used by SAS on long-haul flights) is in the order of one billion SEK (approx 130 million USD). Thus it is of paramount importance to utilize the aircraft as much possible. For short haul it is difficult to sell tickets for nighttime flights, and flights around noon. The peak traffic hours are during the early morning and in the late afternoon. For long haul, the aircraft pretty much operate around the clock. For an insightful text on the economics of the modern airline business, see Doganis [3].

As fuel is also expensive (and several metric tonnes of it are used for each flight, even short ones), flying empty planes must be avoided whenever possible. In practise, only very rare flights to/from facilities where major overhaul is performed, are done with empty planes.

Scheduling of crew cannot directly affect fuel costs or the cost of the aircraft. However, it does influence crew costs, since efficient scheduling might reduce the number of crew needed.

4.1 Stability

Crew schedules should have a property often referred to as stability. This means that the schedule is insensitive to delays. A schedule with very low stability might be a schedule where a crew member is expected to make his/her way across a large airfield in a short time. If his/her aircraft arrives just a bit late, he/she might not make it in time, and the flight he/she should be on will be delayed as well. The potential for this kind of delay propagation should be kept to a minimum.

4.2 Quality of Life

With deregulation and the emergence of low-cost airlines, competition in the airline industry has increased in recent years (again, see [3]). Since crew costs are one of the biggest expenses of the airline, it is natural that it has been one of the first areas in which to try to cut costs, usually by reducing the size of the work force. At SAS, this has led to an increased employee work load, especially for cabin crew. Apart from sheer philanthropy, there are strong economic reasons to make sure that employees are not subjected to unbearable working conditions. In the extreme, of course, pushing employees too hard would make them quit their jobs, and it could also lead to recruitment difficulties. However, there might be more immediate effects.

At SAS, a rather large portion of the Cabin Crew work force does not do any actual work, because of illness. As the cabin crew work load has recently increased, some speculate that there may be a causal relation between increased work load, and illness.

Often people are willing to work more, if they can have some control over when they work. For instance, being free on an own child's birthday might be worth several days of work. In many cases, granting a specific request for a day off might not cost very much to the airline, since people are guaranteed a certain number of days off every month, anyway. When those days occur is not of much consequence to the company, as long as not all employees want the same day off.

All in all, the quality of life of employees is considered important by the airline.

One of the initiatives to bolster quality of life is the "bidding system" used at SAS (and many other airlines).

This lets crew specify what kind of duty they prefer, and when they want to work. The most important types of bids at SAS are:

- Time off (specific dates, week days, or times of day)
- Specific destinations (Bangkok is apparently a popular destination)
- Specific flight numbers
- Length of working periods (few long working periods, or many, shorter?).

Each bid is weighted by the bidding crew member, on a scale from 1 to 200.

There are bids that can be granted many times. For instance, a bid for a specific destination (which can be traveled to more than once in one month).

The bid system should somehow be fair to all users. If the weights were used directly by the optimizer, there would be no value in ever using a weight other than 200. Also, crew could try to game the system by placing equivalent bids (perhaps bidding by destination and the flight numbers that lead to the particular destination). So, some kind of normalization is necessary. This is non-trivial, since it is difficult to know how many times a certain bid can be granted.

The solution used involves a special optimization run, in which a "reference roster" is created. In this roster, each crew member is scheduled as if he/she was the only crew member in the whole company, while maximizing granted bids. In other words, all requests that could feasibly be granted will be granted. If two bids clash (for example, a time off bid, and a bid for a specific flight number during the same time period), the combination of bids yielding the highest possible bid points sum is selected. Some bids, for example a bid for a destination that is not flown by the airline at all, or a non existent flight number, will never be granted. Also, there is a limit that the reference roster must contain at least a certain minimum of production (you cannot place bids to create a reference roster that yields time off every day).

In the production run, the optimizer strives to grant as large a proportion of all bids granted in the reference roster as possible. This proportion is called the bid ratio - the ratio of bids granted in the production roster to bids granted in the reference roster.

An effect of this is, that there is no advantage to be gained by crew in using higher weights - the value of the weights are only ever compared with the weights of other bids by the same crew member. For example, Placing three bids with weights 5,15 and 20 is exactly equivalent to placing bids with weights 50,150 and 200.

To recapitulate, the bid system:

- Crew place bids. Example: “Each New York landing, 200 points. Time off September 15th, 99 points. Time off September 16th, 51 points.”
- The reference roster is created. Let’s continue the example: The free days on September 15th and 16th are both granted, but the New York flight is not granted (perhaps the crew member does not have the necessary qualification). The reference roster now contains 150 points.
- The actual roster is created. Let’s say that the free day on September 16th can not be granted. The September 15th is no problem however. The crew member now has 99 granted points. The bid ratio is 66%.

It is the bid ratio percentage that the optimizer tries to maximize (for all crew members). The rationale being that it is fair if everyone is granted the same percentage of bids.

Chapter 5

Operations Research and Integer Programming

Sometime during World War 2, a bunch of intelligent people realized, that some of the difficult decisions that military commanders and planners faced could be approached from a more formal, mathematical, standpoint than was done at the time. An example might be the question of how close together the aircraft of the RAF bomber command should fly. On the one hand, attacking enemy fighter aircraft will subject themselves to the firepower of more bombers, if the bombers are closer together. On the other hand, the risk of the bombers colliding with each other increases as they are asked to fly closer and closer together. The OR approach to this problem was to use statistics and mathematics to model the problem and calculate an optimum distance between the bombers. An interesting overview of the field of OR, and its place in society, is given by Virginia Postrel [12] (she calls OR "the most influential academic discipline you've never heard of"!).

Among the other problems taken on by the OR pioneers were scheduling problems and logistics problems. For many problems, optimization techniques such as "Linear Programming" (more about this later) are of use.

Operations Research (OR) is defined by Encyclopedia Britannica as being:

"(the) application of scientific methods to the management and administration of organized military, governmental, commercial, and industrial processes.".

Many techniques, such as Linear and Integer Programming, have come to be known as "OR-techniques". There is presently a strong movement to incorporate techniques from other fields of study (for instance AI) into OR, and vice versa.

5.1 Linear Programming

Linear Programming (often referred to as 'LP') is probably best described by example. Here is a scenario:

You are in the grocery store. They sell apples and oranges. The apples cost 25 SEK/kg. The oranges are on sale and cost 15 SEK/kg.

You set out to buy X kg of Apples, and Y kg of Oranges. Your fruit-budget is 100 SEK. This financial limitation means that:

$25X + 15Y \leq 100$

But there's another complication: You are only allowed one bag to carry the fruit in(!), and this bag can only hold 5kg. Therefore:

$X + Y \leq 5$

Now, you're fond of oranges, but even more fond of apples. Actually, your fondness for apples measures 5 units per kg, and that for oranges only 4 units per kg. The question is now, many kg of apples and oranges do we buy to maximize the value of

$goal = 5X + 4Y$?

subject to these constraints:

$25X + 15Y \leq 100$

$X + Y \leq 5$

$X \geq 0$

$Y \geq 0$

?

Above, $5X + 4Y$ is said to be our *goal function*. Now, the joy of consuming an apple or orange may not be easily measured in money, but for many other problems, the goal is to minimize some cost (in monetary terms). In such cases, we talk of a *cost function*. Of course, minimizing a function is the same as maximizing the negation of the same function.

From figure 5.1 you can probably easily see that the optimal solution is:

$X = 2.5, Y = 2.5$

In other words, you should buy 2.5kg of Apples, and 2.5kg of Oranges.

This yields a goal-function value of 22.5. This is an example of a simple *Linear Program* (LP).

The OR terminology can be confusing to people from outside of the OR community. When an OR person talks about a Linear Program, he is referring to the problem that is to be solved by LP techniques. In Software Engineering, the Program is the software system that solves the particular problem it was created to solve. In OR terminology, however, a Linear Program is a mathematically formulated problem of the same type as the Apples and Oranges example presented here.

A definition of a linear program:

- We have any number of variables X,Y...
- We have a number of linear inequalities:

  $a_1X + a_2Y + ... \leq a_{n+1}$

  $b_1X + b_2Y + ... \leq b_{n+1}$

  (and so on, for c, d, e ...)
- We have a linear cost function $C_1X + C_2Y + ...$
- We wish to find values of $X, Y...$ minimizing the cost function while not invalidating any of the inequalities.

In two dimensions, the problem is generally easy to solve. Constraints have the geometrical interpretation of being infinitely long lines, each dividing the potential solution space in two halves, one legal half and one illegal half. A potential solution is simply a point. To see if a point (potential solution) is legal, it is enough to determine if there is any line such that the point is on the illegal side of it. If there is, the solution is illegal, otherwise it's legal. See figure 5.1 for a geometrical depiction of our example LP problem.

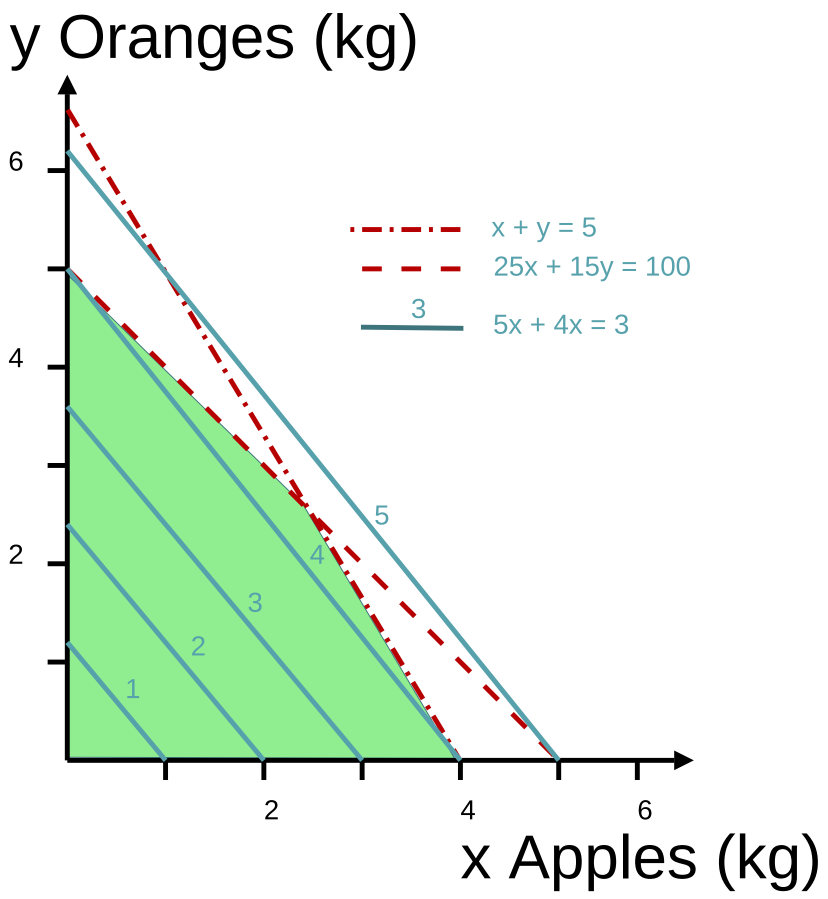


Figure 5.1: A simple example of linear programming in two dimensions. The parallel blue lines represent a contour plot with respect to the goal function (analogous to 'isobars' in meteorology, or elevation contour plots on an orienteering map). They are a representation in two dimensions of the R2->R1 function that the goal function constitutes. The red lines are the constraints to the allowed area. The green area is the legal solution space.

It can be proved, or realized intuitively, that there will be at least one intersection between constraint lines that has a goal function value such that no other legal solution has a higher value, and, which is itself a legal solution. In other words, there may be many optimal solutions, with the same goal value, but one of those is always an intersection between two lines.

One simple way to find a best legal solution, and do it in polynomial running time in the number of constraints, is to find all intersections between lines, filter out those representing illegal solutions, and then select the remaining solution with the highest goal function value.

Generalizing the above example to N dimensions instead of 2 is straightforward. Constraints become hyperplanes instead of lines. And analogous to the intersection of two lines, a point, we have the intersection of N hyperplanes, also a point. These points are called vertices (singular: vertex). The set of legal solutions becomes a convex N-dimensional polyhedron, instead of being a convex polygon as in the 2-dimensional case. It is still the case that one, or some, of the vertices has an optimum goal function value.

Solving general N-dimensional Linear Programs though, is not feasible using the algorithm described for the 2D-case above. There may be far too many vertices to explicitly enumerate them all.

This problem was tackled by George Dantzig, who, in 1947, invented the Simplex method for solving it [8].

1. Somehow find one legal vertex (any legal vertex will do).

2. Find vertices that share an edge with this vertex , in the N-dimensional polyhedron. Call these neighbors.

3. Select a neighbor with better or equal objective function value compared to the current one, and mark it. Marked vertices are not selected again.

4. If no neighbor has a better or equal objective function value than the current vertex, we have found an optimal solution and will stop.

5. Otherwise, we repeat the algorithm from step 2, using the vertex we selected in step 3.

See figure 5.2 for a geometrical interpretation.

The simplex method is said to perform well for many practical problems, but has a worst case time complexity that is exponential in the number of variables. The problem is that the algorithm may, for certain problems, visit all vertices before finding the optimal one.

Note that the description of the algorithm above is slightly vague in step 3. Several neighboring vertices may have a better or equal objective function value.

A 'pivot rule' is a strategy for selecting one of those vertices. Several 'pivot rules' are possible. The actual runtime of the algorithm depends on how we select vertices, on how good our pivot rule is.

At time of writing, it seems no one has found a pivot rule for the simplex algorithm that makes the running-time polynomial in the worst case. A google search for "polynomial time simplex" turns up sentences such as: "there is still no provably polynomial time simplex algorithm". If any researches have actually found a polynomial time simplex algorithm, at least they haven't given it the obvious title and put it up on CiteSeer...

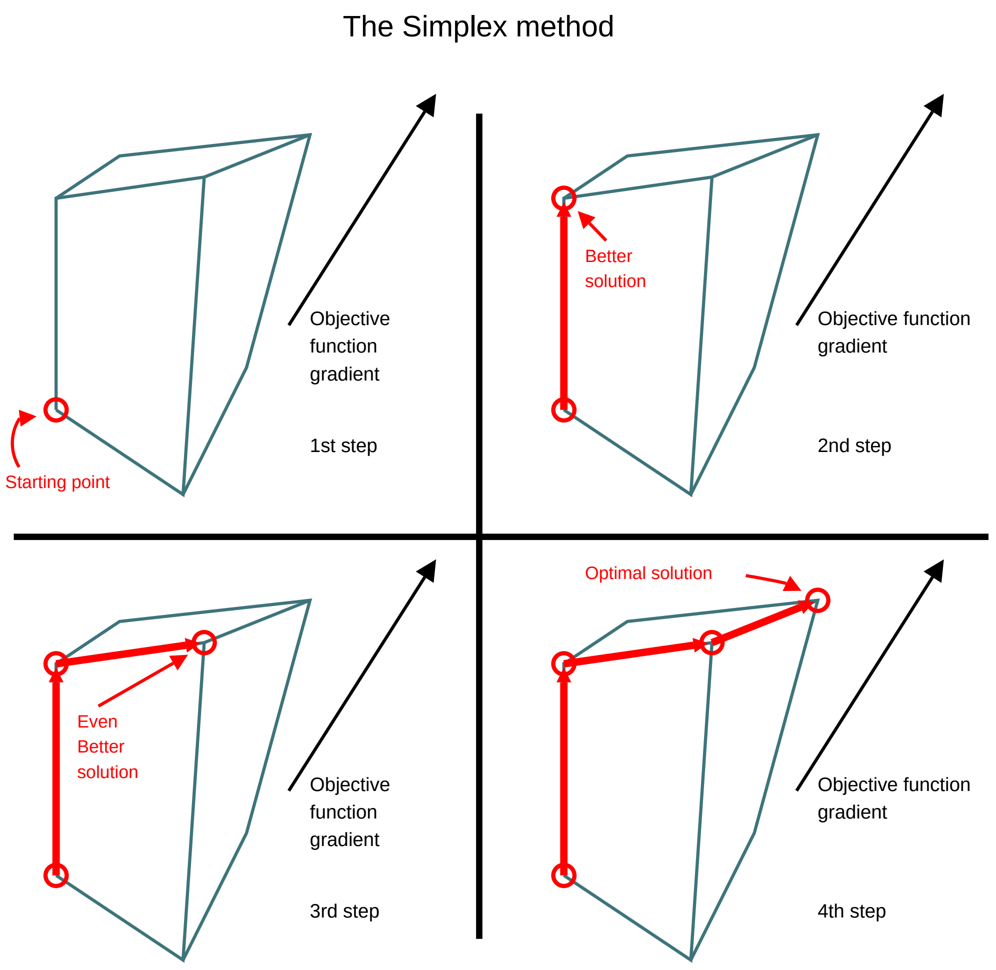


Figure 5.2: The Simplex Method applied on a three-dimensional distorted cube.

However, there are other algorithms for solving Linear Programming problems - so called “interior point” methods, which do run in polynomial time, worst case. For typical problems though, they seem to have similar performance to simplex implementations.

5.2 Integer Programming (IP)

For many real problems (those being solved at SAS, for example), the variables are restricted to integer values. This makes the problem much harder - in fact, NP-hard.

What is typically done is that the problem is “relaxed” - a fancy word for saying that we loosen up some constraints. For instance, initially allowing non-integer solutions. This *linear relaxation* of the problem is then solved as an ordinary LP problem. If one is lucky, the LP solution might yield integral values by chance. If it does not, however, finding the optimal integral values is not trivial.

It may be that simply rounding the variables to the nearest integer produces an optimal solution, but in the general case, the rounded variables might not even constitute a legal solution. Let’s say that the relaxed value of X is 3.4, and Y is 4.7. It might be tempting to say this proves that X is either 3 or 4, but that is not correct. For example, picture a pointed cone shape polytope, with the relaxed solution at its peak, and the only integer solutions near the base (see figure 5.3). There is no general, easy, and quick way to find the optimal integer solutions.

Note that all solutions that are legal in the IP, are also legal in the LP of the linear relaxation (the reverse it not true). This means that the optimal value of the goal function for the linear relaxation is at least as good as that for the optimal IP solution.

Two major approaches for dealing with IP problems exist: Cutting plane methods, and branch-and-bound methods.

5.2.1 Cutting Plane methods

Cutting plane methods start by solving the linear relaxation. Then, a *cut* is generated, which is a new constraint, that does not cut off any integer solution, but does cut off the solution to the linear relaxation (again, see figure 5.3, for an illustration of a cut). The LP relaxation is solved again, this time with the cut-constraint included. If the solution still isn’t integer, a new cut is generated, and the process continues. There is no general purpose algorithm for generating efficient cuts. There is a general algorithm called “Gomory cuts”, using which we are guaranteed to eventually find the optimal integer solution. However, performance is unsatisfactory for many problems. The problem is, that we might end up with an enormous number of cuts, before the solution becomes integral. There are, however, cutting-algorithms that will yield good performance, but only for certain specific problems.

5.2.2 Branch-and-bound methods

The other major approach is to use branch-and-bound methods. These methods also begin by solving the linear relaxation of the IP. Let’s say we get $X = 3.4$ and $Y = 4.7$ as solution to the linear relaxation. Now we *branch*, we make two copies (branches) of the problem, one branch with $X \leq 3$ and one branch with $X \geq 4$. Both of these two branches, both of these two subproblems, are slightly more constrained than the original problem. The idea is to continue branching, until only integer solutions are left.

The set of branches constitutes a search tree. Even though this process is guaranteed to stop, it can generally take a very long time for it to do so.

Let's say the relaxation of the branch with $X \leq 3$ yields an all-integer solution and that it has a lower goal function value than the relaxation of the $X \geq 4$ branch. Even if the $X \geq 4$ branch has a non-integral solution, we still aren't interested in it. We know that none of the possible integral solutions of this branch can be better than the one we already have. This branch has been *fathomed*. This is a clever optimization, that can cut off large parts of the search tree.

Note that, for each integer solution that we find, we can compare it to the linear relaxation of the original problem. We might determine that we will stop the search if we get within a certain percent of the goal function value of the relaxation. This way we can trade some accuracy for a shorter running time.

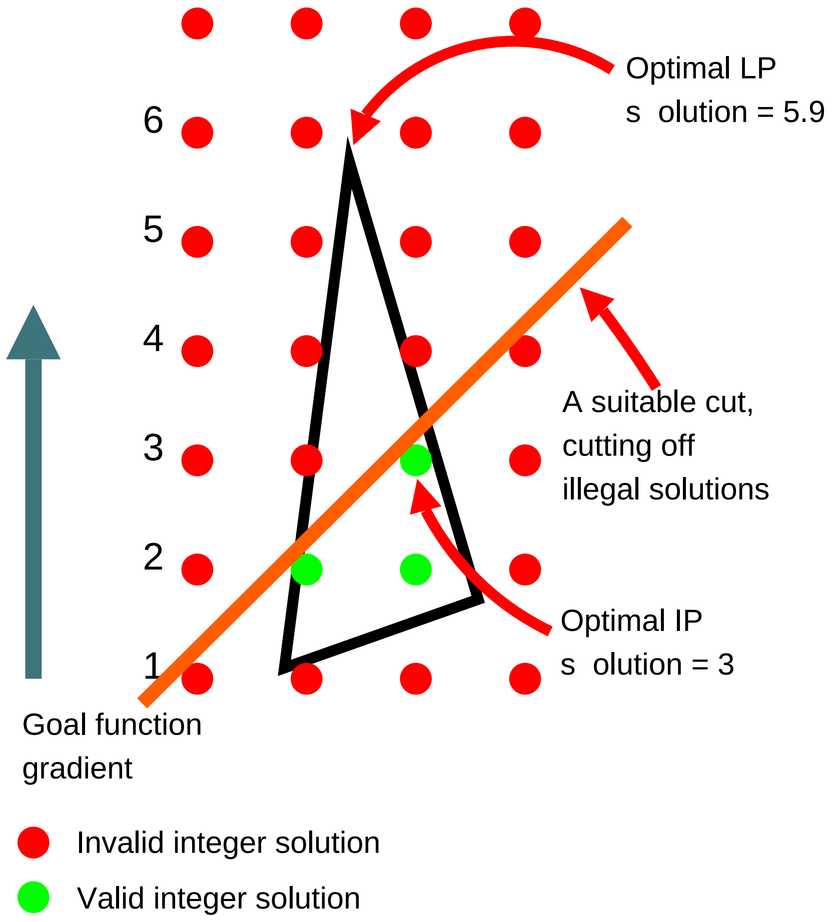


Figure 5.3: An illustration how simply rounding non-integer solutions does not always give a feasible solution. Also, a suitable cut, that will cut off many non-integer solutions, is shown.

Chapter 6

Technology

The pairing problem has been extensively studied within the Operations Research community. Like almost any scheduling problem, the typical formulation of this problem in mathematical terms is (at least) NP-hard. It is also very large, which means that there will usually be no guarantee that the solver will actually find the optimal solution

The pairing problem is conveniently expressed as an Integer Programming (IP) problem, where the columns are pairings (sequences of flight numbers to be flown), and the rows are legs needing crew. It can be viewed as a set covering problem, since all flights have to be covered by pairings (manned). It is very suited to the use of column generators. A mathematical description of this technique is given by Randolph Hall in “Handbook of Transportation Science” [7]. For a more pedagogical description, see “Advanced APC for Pairing” from Carmen Systems [2]. Another interesting document about pairing is “Solving larger crew pairing problems”[9], also from Carmen. Another text about column generation is “Column Generation and the airline crew pairing problem”[6].

The rostering problem seems to have been studied less intensely, but is also usually (at least by Carmen) formulated as a large Integer Programming (IP) problem.

Each column is one possible roster, for a specific crew member. (These rosters have been generated by a kind of heuristic, more about it later.) First, there are rows to ensure that each crew member has at most one roster. Then there is one row for each trip, making sure that each trip is flown by the right number of crew members. Then there may be additional rows for other types of constraints, like ’must fly together’- or ’must not fly together’- constraints between pairs of crew members, or global constraints (more about all these, further down). The task of the IP-solver is basically to choose which of the generated rosters to use. The quality of the generated rosters (columns) is therefore of utmost importance. After one solution has been found, the process starts over, with new generated rosters, and the solutions is improved in an incremental fashion. More about this later.

For a description of the Carmen-approach to rostering, see “Airline crew rostering: Problem types, modeling and optimization” [5].

Figure 6.1 shows the rostering optimization problem in mathematical terms.

An example:
Given:

$$P = \begin{pmatrix} p_{1A} \\ p_{1B} \\ p_{1C} \\ p_{2A} \\ p_{2B} \\ p_{2C} \end{pmatrix}$$

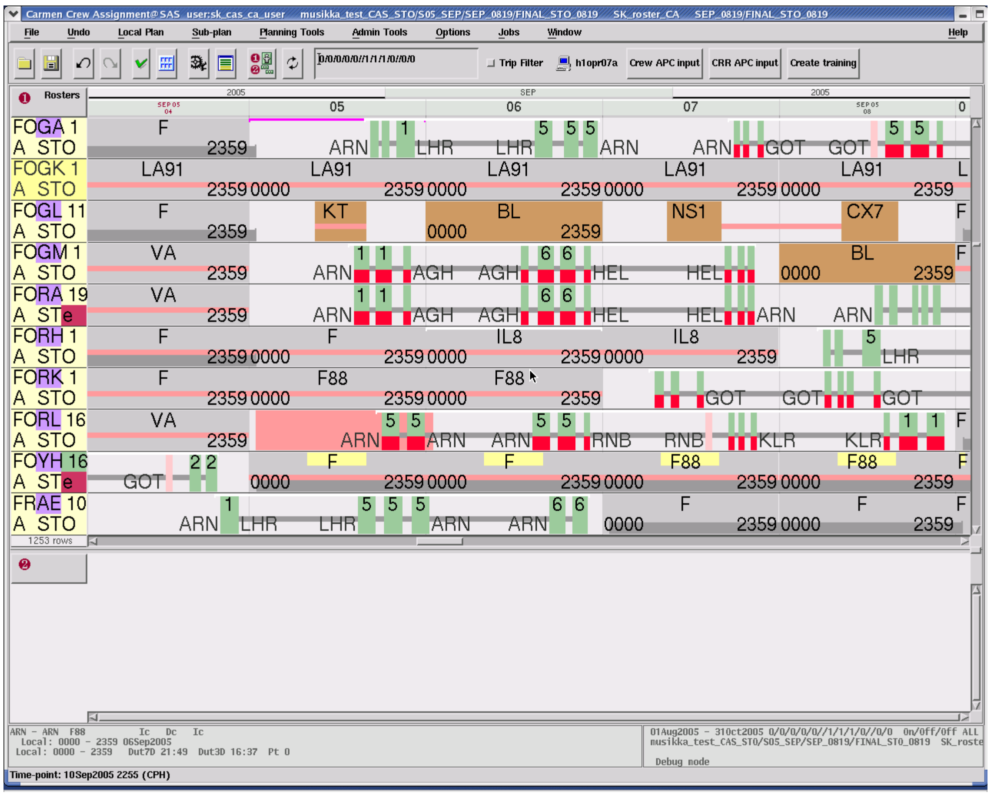
$$C = \begin{pmatrix} 1 \\ 3 \\ 2 \\ 4 \\ 1 \\ 3 \end{pmatrix}$$

Select values in range $0 \ldots 1$ for $P_i, i = 1 \ldots 6$, such that:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix} \cdot P = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

while minimizing
total cost = $P \cdot C$

Figure 6.1: This is an airline with two crew members, “Crew 1” and “Crew 2”, and three pairings A, B and C. Three possible schedules have been generated for each crew member. Each column represents one schedule. The first two rows of the matrix equation ensures that each crew member is assigned exactly one schedule. Each of the last three rows represents one pairing of the three pairings. Each column has a 1 at a row, if the schedule of the column flies the pairing of the row. P_{1A} is a variable that is either 0 or 1. If it is 1, this means that “Crew 1” flies pairing A. The third row makes sure that exactly one crew member flies pairing A. The vector C is the cost of each of the 6 generated schedules. $P \bullet C$ is the dot-product of P and C, and thus the sum of the costs of the two selected schedules.



```
ARN - ARN  F88          Ic   Dc   Ic
  Local: 0000 - 2359 06Sep2005
 Local: 0000 - 2359   Dut7D 21:49  Dut3D 16:37  Pt 0
```

```
01Aug2005 - 31Oct2005 0/0/0/0/0//1/1/1/0//0/0  On/Off/Off ALL
musikka_test_CAS_STO/S05_SEP/SEP_0819/FINAL_STO_0819  SK_roste

Debug mode
```

```
Time-point: 10Sep2005 2255 (CPH)
```

Figure 6.2: Carmen Rostering software (CAS) in action.

6.1 The Carmen Software

SAS uses software from Gothenburg-based Carmen Systems AB. Relevant to this work is a piece of software known as CAS, used for crew rostering.

CAS allows the user to manually assign trips to crew members. CAS will also tell the user if he/she accidentally creates a schedule that violates rules. See Figure 6.2 for a picture of the user interface of CAS. For information of modeling in CAS, see 7.

CAS has a built in optimizer, that will try to create an optimal schedule with respect to the specified cost function, while not breaking any rules. There are many parameters that may be set, to allow fine tuning of the optimization process. One of the components in the optimizer is an IP-solver. CAS uses a solver from Dash Optimization.

The optimizer works in an incremental fashion. It starts with one *construction*-operation, followed by many applications of an *improvement method*. The objective of the construction operation is: Given a set of crew members, and a set of pairings, create any valid solution. This step does not generate an optimal solution, but rather is a starting-point for later steps. A *chain* is the sequence of flights a specific crew member is to fly during a certain period - it is simply another word for a person's schedule. Chains are created left-to-right; meaning, for each crew member, pairings occurring later and later in time are successively appended to the chain. After each pairing is added, legality of the entire chain is checked.

After an initial solution has been created, a time-window is moved across the roster. The window is approximately one week long, and is moved in increments of 48 hours. For each increment, all pairings within the time window are deassigned (made uncrewed). Then, a couple of different possible chains are generated for each crew member. The IP optimizer later selects one chain for every crew member, out of the set of per-crew possible chains, see Figure 6.3.

This generation of chains (to be selected by the IP optimizer) greatly affects the outcome of the optimization process.

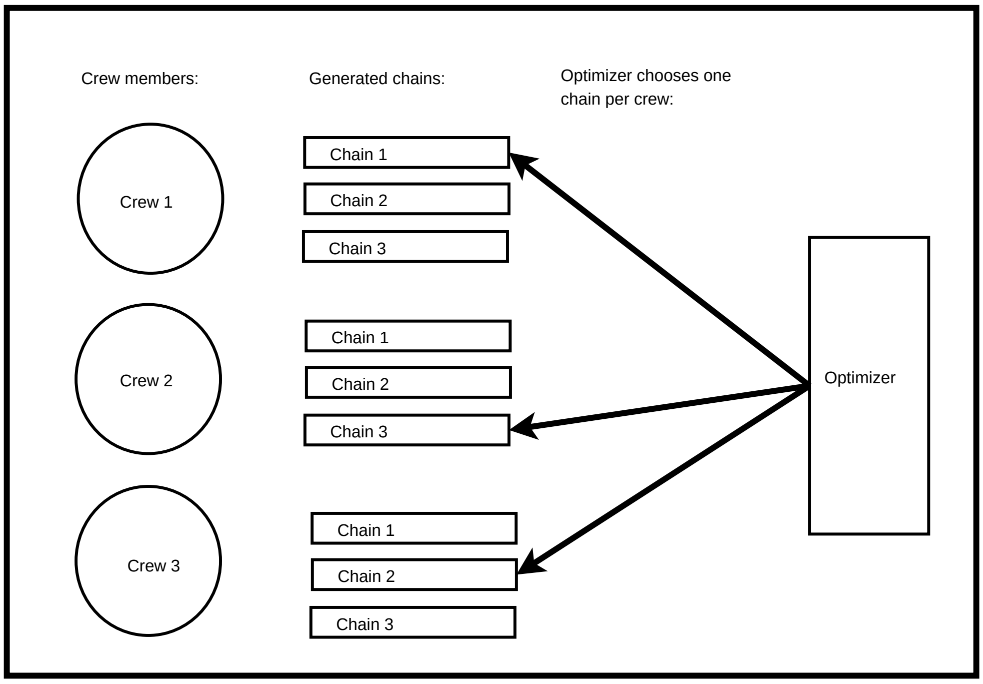


Figure 6.3: An illustration of the optimization process.

6.2 Other approaches to rostering

Other approaches to both pairing and rostering have been proposed, the most interesting perhaps the technique of Constraint Programming. Caprara [1] use ideas from both constraint programming and traditional IP for crew rostering.

Genetic Algorithms (GA), Neural networks and Simulated annealing methods have also been proposed. Ideas from GA and simulated annealing are in fact used in many 'IP'-based methods - that is, the idea of incremental refinement, and the idea of using some randomization method to escape local cost minima. The problem with pure GA and Simulated Annealing (whatever that may be) is of course one of performance. Both approaches will theoretically generate high quality schedules, given enough time. Without efficient heuristics, however, the required time may be in the order of the expected age of the universe, on today's computers. As the heuristic gets more refined, a shorter runtime results. This usually means that the method is getting closer and closer to being a completely problem specific technique.

Chapter 7

Modeling

In order for CAS to know what the rules are, the rules have to be specified. This is done through programming in Carmen's own "RAVE" language. Rave also allows the specification of a cost function. Rave is a non-Turing-complete state description language. See Table 7.1 for an example of some Rave code.

```
%block_time% =
    arrival - departure;

rule max_five_hour_blocktime =
    %block_time% < 5:00;
end
```

Table 7.1: Some Rave-code that would entirely prohibit flights longer than five hours.

7.1 Finer points of the Carmen Optimizer

During the years, extensive tuning work has been carried out by Carmen. Examples of this are:

Path Graph Some pairings cannot occur in the same chain. The simplest case is pairings which overlap in time. But there are also other concerns, such as rest time rules. This realization makes it possible to create a DAG (directed acyclic graph), with pairings as nodes, and an edge between any two pairings which can occur in the same chain. Using this graph, one can save many evaluations of the rave rules. Since evaluating a chain with rave is computationally expensive, this is a good thing.

Trip sorting When generating chains, the solver selects pairings to include. By default, during construction, the nearest pairing that may be flown is always selected first. If Trip Sorting is activated, the user can guide the generator in its selection of pairings. This guidance is given through Rave. This can make the solver find higher quality solutions using fewer iterations, even though each iteration is likely to take longer.

Illegal Subchain problem Since the optimizer evaluates legality after each added pairing, and since it discards illegal solutions, it will never choose a pairing, which makes the chain constructed so-far, illegal. In other words, no sub-chain of a desired full chain should be illegal. An even more concise description is that *no legal chain may become illegal by subtracting pairings from it*. This requirement allows considerable speedup of the generation process, since illegal combinations of pairings can be rejected quickly.

7.2 A suitable metric

What is a good schedule? It is desirable to minimize the cost of implementing the schedule (it should require as few air-men and -women as possible). It is also desirable to maximize the expected income during the schedule period. It is not immediately obvious that crew scheduling affects company income, but it does. Tickets for canceled flights must typically be refunded, also, otherwise recurring customers might not return if dissatisfied because of a delay.

It is of course nice if the best quality metric that we can create is used directly as cost-function for the optimizer. A way to evaluate the result of changed rules or cost function, is therefore the following:

1. Run optimization using old rules and cost function.
2. Load new rules and cost function, and check cost for the created solution.
3. Run optimization using the new rules and cost function.
4. Check cost. Calculate how much cheaper the solution is now, compared to the one in step 2. This difference is precisely the amount of money that SAS has saved (provided that the cost function is correct).

Unfortunately, the cost function presently used at SAS is rather large, and difficult to grasp. Also, some of its terms are very artificial, putting a cost on things such as “too much rest”. It seems unlikely that any crew member ever has complained about having too much rest. One might argue (be it correctly or incorrectly) that such artificial costs can guide the optimizer toward more desirable schedules. In the case with “too much rest”, the case is that if one crew member has an excessive amount of rest, that means some other crew member possibly has too little rest. However, an alternative way to model this is to put a specific cost on the latter.

A few key performance indicators/key properties have been identified. They will not be weighted against each other, since that is difficult to do (see section 11). Instead, each schedule will be evaluated based on all the indicators. They are:

Unmanned flights Any schedule that does not manage to assign crew to all (or at least almost all) flights, is unacceptable.

Fairness across categories of crew Ideally, crew members who have a 75%-part time employment should work 75% as much as those who have 100% (full time) employment.

Fairness across individuals All crew members within each category should ideally work equally much. All 100% employees should ideally work the same number of hours each month.

PBS bid satisfaction Crew can place “bids” to work on a specific flight, or have a specific date off. These bids cannot always be granted. High bid satisfaction (high ratio of granted bids to total (feasible) bids) is desirable.

Proximity to minimum rest time rule How close to breaking the minimum rest time rule are we?

Proximity to 90p rule How close to the “90p rule” are we?

Proximity to 270p rule How close to the “270p rule” are we? To what extent does the schedule push the limits of government rules and those of the union agreements?

Aircraft qualification problems Crew must fly each aircraft type which they are qualified for, at least once every 90 days. If they go 90 days without flying a specific type (Boeing 737, McDonnell Douglas MD-80, etc) they must receive training. First they need a one-day course, and then they need to fly two ’release flights’. The release flights are regular production flights, with passengers, and must be crewed normally, in addition to the crew member in training. Thus, each time any crew member does not fly one of the aircraft types he/she is rated for, two production days are lost. Therefore, it is desirable that each crew member is scheduled to fly each type often enough.

Check-in times It is considered unpleasant to start the first day after a free period very early or very late. Specifically, crew does not enjoy checking in before 06:00 in the morning, or after 22:00, the first day after vacation or free time.

Blank days Since the blank days are in effect used as standby, and since standby need is calculated by experience, a drastic reduction in the number of blank days in a schedule could be a problem.

7.3 Softening a rule

Sometimes a distinction is made between *soft rules* and *hard rules*. Hard rules must never be broken. Soft rules are rules which can be broken to a degree, but breaking them incurs a cost. Usually, there is some limit to how much a soft rule may be broken, after that limit is reached, it behaves like a hard rule. Soft rules are thus implemented with a term in the cost function, as well as a hard rule.

Softening a rule is the act of replacing a hard rule with a soft rule. This may be attractive, since it increases the granularity of the cost function.

Three (very similar) softenings can be done:

1. The rule cannot be broken, and actually, there should be a cost to even coming near.

2. The rule could really be broken by a small margin. It’s just that it would cost. Breaking the rule too much must still be illegal.

3. The rule could really be broken, it’s just that it would cost. No amount of breakage would be illegal.

All these can be unified to nr 1. For nr 2, it's just a question of adjusting the cost function, and moving the rule so that it is placed properly at the limit of legality. For nr 3, it's a matter of moving the legality-limit so far away that it is never reached. So, mathematically, these three are all the same.

7.4 What it should cost

The general guideline for choosing cost-functions is clear: the function should somehow reflect the real impact on company profits that each point in its domain set actually contributes. This would work beautifully if each item that a cost-function was associated with was directly mappable to a part of the income or expenses. Put differently: Company income is only decomposable to different contributions down to a certain degree. Sure, it might be possible to calculate how much income a certain flight generates, but how about each pilot? Or each deadheading passenger? And for that matter, part of the expense for the airline are "fixed costs", such as headquarters, planning etc. The cost for these should somehow be subtracted from the profits of each flight. What is typically done in real life is that part of the cost for having the headquarter is allocated to each flight that you operate. In the same manner, you could allocate income to the different participants in a flight. Without a captain you can't fly, and lose all income, so the cost of *not* having a captain should be as large as the income of the flight. But the copilot is also necessary, so the cost of him not showing up should be the same. But according to this model, you'd lose twice the total income if both pilots where unavailable. This erroneous conclusion is the result of the "true" global cost function not being decomposable into a term for each pilot (captain and co-pilot). In other words, there's a tendency to "Count things twice".

One thing you can do for any (even non-linear or non-continuous) multi-variable function, is to keep all but one variable fixed, and thus create a function of one variable, that gives the change in the cost function as that one variable is changed. This is what we're going to do.

The reason that we do not have one single multi-variable cost-function, that is a function of all relevant parameters within the problem, is in part that the Carmen optimizer imposes the (artificial) constraint that the cost function be a sum with a contribution from each crew member, with only a few exceptions. (The most important exception is that there may be a contribution from unassigned flights.) The reason for these limitations is purely that it makes the job of the optimizer easier.

In actual planning, the point of view presented in this section is hardly very useful. And most costs for rostering are simply decomposable into a cost per crew member, so a less formal, more intuitive point of view will tend to work well. Most of the more complex issues are outside the designated area of responsibility for the planning department anyway.

The only practical approach is to try and identify bad properties that a schedule can have, and, using experience, intuition, and case-by-case specific reasoning, try and estimate what the cost of the property might be, given that all other parameters have 'standard values'. Many bad properties are such that you strive to avoid them altogether. The "count things twice" problem above then goes away.

7.5 Finding rules with coarse granularity

An experiment was made to find coarse grained rules which might be suitable for elaboration. The optimizer was run with all rave-rules turned off. The only constraints placed on the schedule in this mode is that people may not be scheduled on two activities simultaneously, and that each activity must be start at the same location as the previous one ended.

This produced an extremely tight schedule, where crew worked approximately 17 hours/day. Then the rules were turned on again, and the software was asked to list all rules that were broken. The idea being, that rules not even broken in this schedule, probably do not affect scheduling much.

After filtering out rules which are never broken, and rules which regard rare occasions (such as special training, rules regarding start of vacation, late check-outs etc), the following remain:

Rule	Description	Benefits from finer granularity?
max 270 duty points in 168 hours	A crew member can only collect 270 duty points in any 168 hour period	Yes, we want to avoid even being *near* the 270p limit.
Minimum two freedays after duty	A crew member must always have at least two whole calendar days off after each sequence of work days	Yes, this is a very limiting and inflexible rule.
Minimum rest	A crew member must have at least a total of 10 hours of rest after each work day	Yes, we want to avoid being near this limit.

Table 7.2: Rules that were identified as candidates for increasing their granularity.

7.6 Two ways to set costs

One way to set costs is to try different costs until the volume of the undesirable parts of the schedule decreases to a desirable amount. Let's call this an artificial cost. It is artificial in the way that this cost function isn't providing any new insight. Another kind of cost-function is when you can quantitatively decide with some certainty what the undesirable property actually costs. One example of the latter might be well-defined taxes and fines, (measurable) resource inefficiency and the like. This second kind of cost provides us with new insight into the problem. A made up example of one possible such insight might be a courier firm that realizes that its employees should intentionally park in no-parking zones, and simply pay the fines, rather than have to park further away, and thus provide slower service to their customers. The example given in section 8.1, about the minimum rest time rule, belongs to this second class of non-artificial costs.

7.7 Softening a rule: Designing the cost function near the hard legality limit

So how much should it cost to come near a legality boundary?

A proposed method to determine cost:

1. Pick a rule, and a value, X, which is a measure of how close we are to breaking the rule (making the chain illegal).
2. Find the possible effects of X coming close to the limit of legality.
3. Find the probabilities for the different possible effects.
4. Find the cost of each effect.
5. Calculate the expected cost Y of each effect - the product of probability and cost, as a function of X.
6. Sum all the expected costs Y.

This will give the cost Y as a function of X.

7.8 Other reasons to use soft rules

The reason for using soft rules is usually cited to be the inevitable uncertainty of (airline) operations. If there were never any delays, most soft rules would be unnecessary.

Sometimes, however, soft rules provide a way to avoid putting an arbitrary limit for rules where there is no actual fixed limit. For example: People dislike getting up early in the morning - but before which time in the morning is “early”? This is actually an instance of the Sorites Paradox, and might be a good instance for using soft rules. (One classical example of the sorites paradox is: Everyone agrees that a man with less than 5-10 strands of hair is bald, and that a man with 100000 hairs is not bald. Everyone agrees that one single hair less or more does not cause a man to be bald or not bald. Then, by induction, all men are bald, and all men are not bald. See [10] for a nice text about this subject within philosophy.)

There’s a high level of non-transparency to soft rules, however, they’re difficult for people to gain insight too. If the union agreement says “no one should be forced to start work before 06:00”, then this is easy for everyone to understand. It is also easy for the Union to monitor if the rule is being honored by the airline. However, if the union agreement says “no one should be forced to start work before 06:00, unless the company can save at least 10 SEK per minute earlier that they start”, it is not very valuable to the crew/Union. They will have a hard time monitoring if the rule is being honored. One possibility would be if the person would get (part of?) this sum of money as a bonus, but this will be difficult (but perhaps not impossible) to implement in SAS, because of a strong tradition of fixed monthly salaries. One possible action is to award extra free days, free time, in response to undesirable properties of the schedule.

Another possibility is to treat before-06:00-minutes as a resource separate from money, and guarantee that no more than, say, 120 such minutes is to be scheduled per month. This is easier for the Union to evaluate, but it is unfortunately not too good from an optimization point of view.

It cannot be modeled conveniently as an extra cost term in the cost-function. While it is possible to find a cost-per-occasion that, for a certain schedule, yields 120 occurances, there is no reason that the same cost-per-occasion will hold for next month's schedule. Tuning *one* such variable each month might be doable, but more than a few is not reasonable (the "Curse of Dimensionality"). In general, it is quite possible, or even likely, that the costs will be coupled, modifying one affects the outcome that the other is to govern.

Constraints such as 'max N occurances of feature X' can be modeled in CAS by what is called "global constraints". Carmen explicitly recommends that such constraints be avoided, for performance reasons.

Chapter 8

Planned Experiments

This section discusses experiments made by the author of this paper.

8.1 An application: Minimum Rest Time

This section gives an example of how the method described in 7.7 can be applied to the "Minimum rest time" union agreement rule (see table 7.2).

8.1.1 Step 1: Picking the rule

There is a rule that says that there must always be at least 10 hours between block on to block off (10 hours from end of flying duty to start of next flying duty) at non-base station, and 12 hours when on base (i.e. Arlanda, for SAS Sweden).

8.1.2 Step 2: Possible effects

There are a few problems associated with not complying with this rule:

- The Union agreement says that SAS must pay each crew member that does not get enough rest an amount equivalent to 11.84% of their monthly salary, per occasion. This cost is of course known.
- Apart from this, a crew member that, because of delays, does not get the minimum rest, has the right to be rescheduled. The cost of this is much more difficult to estimate. We must know what percentage of crew actually demand rescheduling, perhaps as a function of how badly they miss the rest target. We must also know what the cost of rescheduling someone actually is. It will require the use of standby, or that someone else is paid to come and work in their spare time. The cost of using standby is pretty clear - in the long run, standby personnel are personnel that could otherwise have been used for something different. Additionally, at SAS, there are limits to how much standby a crew member can be assigned to (standby is not considered attractive).

8.1.3 Step 3: Finding probabilities

Let X be the margin we have against violating the rule. If X is 15 minutes, we have planned a schedule so that the crew member gets 12h15min rest on base, for instance.

The probability of not getting enough rest is a function of the margin, X. X can be seen as a stochastic variable. Its distribution is difficult to know exactly. One approach is to look for actual traffic data from a previously flown month, and collect lateness statistic for each flight in that month, into a histogram. See figure 8.1 for an example of such an histogram. If a certain flight had the same probability of being X minutes late, each month, this histogram could have been used to make predictions on how late flights are likely to be in the current month.

Now, the distribution is not the same each month, even though the distributions in different months do correlate to some degree.

The correlation coefficient for the average lateness of a randomly selected flight in August 2005, and the average lateness of the same flight in September 2005, was 0.42.

Thus, information from previous months can give *some* information regarding how likely a flight is to become more than a certain number of minutes late.

In more mathematical terms, the correlation calculated was the Pearson correlation coefficient between all X_f and Y_f, where X_f is the average lateness in August of the flight with flight number f, and Y_f is the average lateness in September of the flight with flight number f. This correlation coefficient can then be interpreted as the amount of information regarding the lateness of a specific flight in the next month, that we get from knowing how late it was in the current month.

A possible criticism of this calculation is that the average lateness of flights, with the flight number picked randomly, probably does not follow a normal distribution. Even so, the correlation coefficient does convey some information about the covariation of X_f and Y_f.

All in all, one can at least try to use what little information about the expected lateness of different flights we have, to estimate how likely we are to break the minimum rest time rule. It will probably be better than not making any estimate at all.

8.1.4 Step 4: Find the cost of each effect

The cost of allocating standby varies according to the seasons. It also varies from day to day. It is impossible to model exactly. For a certain day, it might be that standby personnel were in fact available, but were not used. Since SAS personnel have a fixed monthly salary, it would not, in such a case, have incurred any cost to use the standby crew. To simplify matters, let us just assume that the cost of using a standby crew member, is simply the cost of one working day, for an average employee with the right qualifications. Since this is pretty close to 11.84% of the monthly salary (there are only around 17 working days per month), an obvious and attractive simplification is to not differ between the two different possible effects.

8.1.5 Step 5: Calculate the expected cost Y

For each flight, for each crew member, we look up the probability of breaking the minimum rest time rule in the histogram for the planned margin. We then multiply this probability by the cost, as determined in step 4.

8.1.6 Step 6: Sum all the expected costs

We sum the contribution of all flights for each crew member. We then sum the contributions from all crew members.

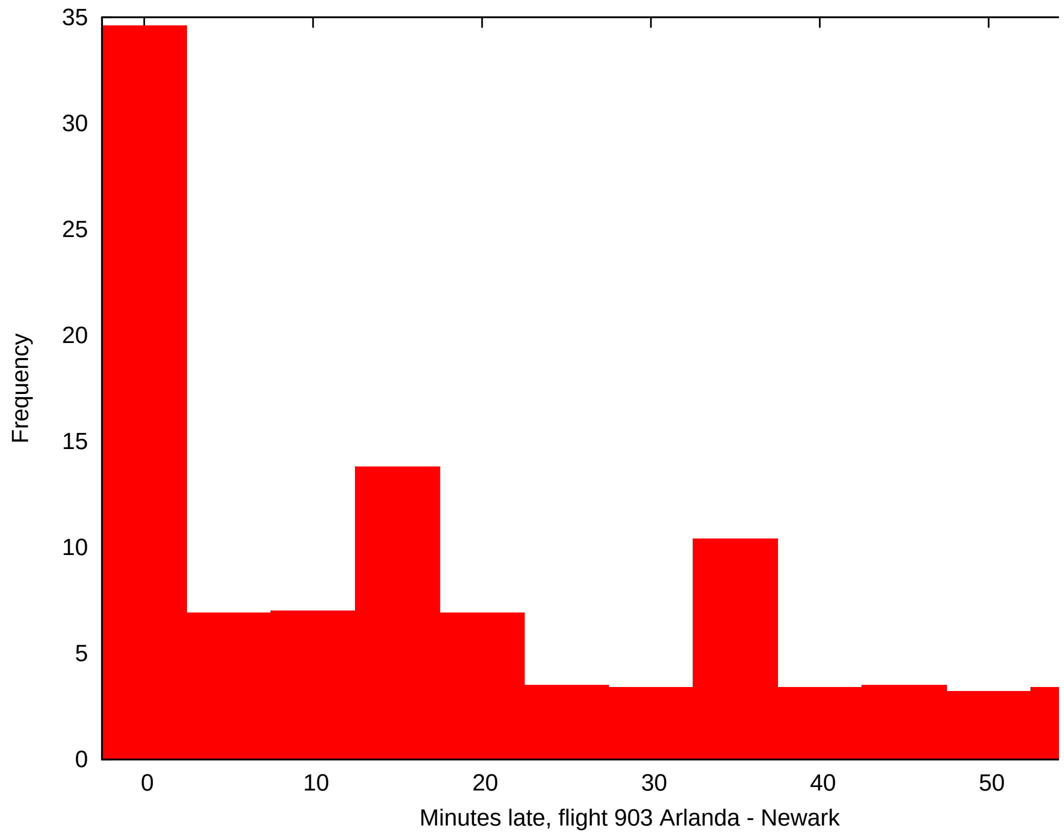


Figure 8.1: An histogram showing the probability of a delay of a certain number of minutes, for flight 903, during September 2005, in 5-minute steps.

8.2 TP - variable free time after duty

SAS crew have a monthly salary, which is independent on how much they actually work. However, they are guaranteed a certain minimum number of off-duty days every month. At present(January 2006), the formula is that each crew member must have at least 12 whole days off every month (for instance, 6 two-day periods, or 4 three-day periods). However, to count against the 12-day rule, off-duty days must be awarded in consecutive two-day periods. The counted unit is calendar days, not arbitrary 24h periods. For instance, it is not allowed to have duty on early Monday, not on Tuesday, and then late on Wednesday again. If such a schedule is created, Tuesday has to be considered an on-duty day as well, even if the free time period is more than 48h long. Therefore, it’s not acceptable for SAS to create schedules with single days without work. If SAS does this, it has basically given the crew member a free extra day off. This limits the schedulability of crew. These rules can be found in the Cabin Crew Union-agreement (Swedish: kollektivavtal), known as “Gula boken” [11].

Complaints have been made by crew that the workload is sometimes excessive. One proposed idea is that more free time could be awarded after heavy working periods (WOP:s). compared to light WOP:s. To do this we need a measure of what constitutes a heavy WOP. Conversation with experienced crew planners at SAS has yielded the following picture:

Length of duty Probably the most important factor.

Number of legs A high number of connections is likely to cause stress.

Charter flights Many airhosts/hostesses consider charter flights to be tough. These flights have a higher frequency of unruly/drunk passengers, and of passengers unaccustomed to flying.

Repetitive work Crew typically do not enjoy to fly to the same destination over and over again.

Meal times It is typically unpleasant to have meal stops at unorthodox times of the day.

Disruption of circadian rhythm Non-regular sleep times have been determined to have a grave impact on health.

These criteria have to be weighted, to be usable by the solver. This process will not be described in this paper. It is a process that, to have any chance of being effective, has to proceed in close cooperation with management, union representatives and crew.

8.2.1 TP

The workload is measured in units called TP (an abbreviation of two Swedish words, meaning approximately “Heaviness Points”). Each WOP (Working Period = WOP) produces number of TP. Free time is awarded after each working period as TP/8 hours. So, if a particular WOP generates 400 TP, the crew member will have 50h off after it.

The TP system replaces the rule that all free time must come in chunks of at least 2 whole calendar days.

The model used for calculating TP is given in table 8.1. This model was developed after conversation with experienced crew planners, and with a small sample of air hostesses. It should be seen as a starting point for discussions with fatigue-management experts, Union representatives and other scheduling stake holders. Such further refinement of the model is outside the scope of this report, however.

It would have been desirable to include rules for bad meal times, but sadly, this isn’t really possible, since meal-times are generated *after* the optimization phase, as a post-processing step, and meals will be included where they can fit. The logic of the post-processing step could of course be reproduced in the optimization step, so that we could second-guess the post-processor. This was deemed to be a too large effort for this thesis work, however.

Also, repetitive work was not modeled, because of prioritization and limited time.

The last two rows of table 8.1 might need a bit of explanation. The idea is that it is more strenuous to work when you have already worked very much the same day, or the previous days, respectively. Thus, we want to give extra TP for long days, and for long WOPs.

Item	Volume	Cost
Duty hour, daytime	/h	6
Duty hour, nighttime	/h	8
Each leg	/leg	5
Charter leg bonus	/h	3
Check in more than 30 minutes earlier than the day before	/exceeding hour	6
Check out later than 36 hours after check-in the day before	/exceeding hour	6
Too long day	X = Each TP from duty hour or landing, exceeding 40	$\left(\frac{X}{9}\right)^2$
Too many TP in one WOP	X = Each TP exceeding 300	$\left(\frac{X}{35}\right)^2$

Table 8.1: The definition of TP, in numerical terms.

Since it is difficult to say exactly how much work is "a long day", we want a function that gives little contribution up to a threshold, and then successively a larger and larger contribution. In mathematical terms, $k \times (u(x) \times x)^2$, where u(x) is the heaviside function, and k is an arbitrary constant, seems to do nicely. In the table a less mathematically strict definition has been used.

8.2.2 Max 1000TP per WOP

Using this model for TP, some working periods will generate excessive amounts of TP. This is not because of a fault in the TP model, but because some activities within the SAS scheduling environment are presented in a way that is sometimes misleading. For instance, when a person has office duty, this is usually listed as being from 0:00 to 23:59 each day, every day in the week. In actuality though, the person is only working normal office hours, and only in the week days. Because of this, and a few other similar factors, a limit is necessary on how much TP a certain working period can generate. This limit has been put at 1000 TP per WOP.

8.2.3 Two phases

To minimize risk in the implementation project, it was decided that the TP model would be implemented in two phases. The first phase would use a very simplified TP model, one including only duty time. The second phase would implement the full-blown model in table 8.1.

Phase 1 Only duty time contributes to TP.

Phase 2 All elements in table 8.1 are modeled.

8.2.4 Cost of TP

It is not absolutely necessary to assign a cost to TP. Having a cost would tell the optimizer that it should try to minimize the amount of TP. For Phase 1, however, this is not possible. Since it only considers duty time, and since the total duty time is known beforehand (all flights are to be manned), the total Phase 1 TP amount if also known beforehand. For Phase 2, on the other hand, the total amount of TP is not fixed. For example, by arranging pairings in a clever way, the optimizer can minimize the disturbance of the circadian rhythm of crew.

This makes it possible to minimize the amount of TP awarded. To make the optimizer actually do so, it must have an incentive. Therefore, a cost per unit of TP is a good idea. This cost should not be too large, since then the optimizer will shift too much of its focus toward minimizing TP, and less to other costs. A cost of ca 0.2 per awarded TP brings the cost into the right order of magnitude. See 8.2.7 for a related extra component to the cost function.

8.2.5 Too long days/Too many TP

The rationale for the two “too many TP” contributions (the last two rows of the table 8.1) is that it is more fatiguing to be working hard (earning many TP) when you are already tired.

For instance, on the shorter time scale, when crew have already earned 100 TP one day, and are very tired, each additional TP they earn is probably more fatiguing than the first.

On a longer time scale, a really heavy day after you have already had an arduous working period, is probably perceived as more burdensome than the same day would have been at the start of the working period.

8.2.6 Fairness

’Fairness’ involves distributing duty in a fair way between different crew members, and different categories of crew members.

There are many aspects of fairness between crew. One of the reasons for trying to achieve fairness is to keep crew happy. Therefore, the *perceived* fairness is what is interesting. There are many measures which can be taken on the schedule. We have the number of hours worked, the number of long-haul duties, the number of free days after long haul, the number of early check-ins, the number of charter flights, the number of other ’heavy’ work days, and much more. It is understandable, that any crew member who (or group of crew that) feel(s) that *he/she/they* are the ones who have to get up early every morning, fly 5 legs/day for an entire week, or hardly get any spare time after LH, will feel that they are being treated unfairly. It is easy to imagine that the schedule receives a lot of attention from crew. After all, the schedule does in some aspects control their lives at work. It seems likely that the total amount of thinking about the schedule done by crew, is larger than that done by the people at the planning department. After all, there are over a thousand crew members, but only 30 or so crew planners.

All in all, it is difficult to achieve fairness.

It is impossible to achieve fairness over a short time span. For instance, it is impossible to give all crew an equal number of long-haul trips in a one-week period. There simply aren’t enough such trips for everyone to get even one. Over a four week period, however, it might be possible. Now, trying to achieve fairness over, say, number of duty hours and number of early mornings as well, might be difficult. Different long-haul pairings have different number of duty hours, so by assigning the number of long haul pairings in a fair way, the work to give a fair distribution of duty hours has been made more difficult. Let's say it's still possible to achieve fairness with respect to duty hours. Now since those with low-hour LH will need high-hour short haul, and since high-hour short haul might start earlier in the morning, this might impact the fair distribution of early mornings. This kind of reasoning is extremely difficult, and is included here just to give some insight into how different aspects of scheduling intertwine. The only effective way of making predictions of the effects of different constraints on a schedule, is to do an optimization run. Experiments have shown that any constraints to scheduling (such as fairness) will lower productivity, lower PBS bid satisfaction or otherwise affect the schedule negatively. Fairness becomes simpler to achieve over a longer time period, though.

An alternative to trying to achieve fairness of multiple parameters, is to somehow weigh these parameters against each other, and then distribute the weighted sum in a fair way. The most straightforward way, in turn, of weighing parameters against each other is probably to express them in a sort of common currency. The TP outlined in section 8.2.1 is one such currency. The question whether a common currency is suitable, or if each parameter should be distributed on its own, is one whose answer depends on the situation. Apparently, in the world economy, each single currency - such as the US Dollar - seems to be usable for determining the distribution of an enormous number of different types of resources. Between people with similar income, few people complain that the distribution of goods and services is unfair. Admittedly, TP and the Dollar work quite differently. Dollar are desirable, while TP are something you want to avoid. However, the negation of TP works quite similarly to the Dollar. If the TP model is correct, if the trips are priced correctly in the TP currency, and if TP are distributed fairly (each crew member receiving an equal amount of TP), it seems likely that the work load would be considered fair, by crew.

And it is much easier to distribute TP fairly, than to distribute each of its terms fairly.

8.2.7 Mechanism for fairness with respect to TP

There are basically two ways to formulate a fairness constraint: You can either use a rule, or a cost.

For the first approach (using a rule), we start by realizing the following: Saying that all crew members should have an equal amount of TP is equivalent to saying that no crew members should have more than the average number of TP. Absolute fairness (all crew having equal TP) is unlikely to be achieved in practice. A relaxed fairness criterion for a schedule is the following:

- No crew member should have more than N TP in the schedule.
- N is only a little larger than the average TP value of all crew in the schedule.
- Flights should be adequately manned.
- N should be as small as possible, while not violating any of the above constraints.

Of course, finding the optimal value of N is not trivial. One approach could be to first do a run with a high value of N, then a low one, and do a kind of binary search for the optimum value. This will require many optimization runs, each of which takes around 10 hours (wall clock time). The situation is not as bad as it might sound, however, since the optimum value of N typically does not vary by more than a few percent between schedules, and might not have to be recalculated for acceptable fairness to result. Also, a helpful rule for guessing N is that it must be larger than the average TP.

The other approach, is to use costs. The simplest approach is to create a new term in the cost function for each crew member. This term is given the shape TP^2. Now, TP will have to be distributed evenly to minimize cost. Again, the fact that the total amount of TP is not constant does undermine this reasoning to some degree, but in practice, TP vary little enough for this approach to work. One problem, however, is that using a cost function, by the very nature of the concept of costs, will allow *some* crew members to have really bad schedules, if that makes many crew members get better schedules.

Let's say we have two schedules each with 100 people, A and B. In schedule A all 100 work at a rate of 100%. In schedule B, 99 people work at a rate of only 80% and one poor soul works at a rate of 200%. Now, the latter schedule is hardly more fair, but the criterion given in the previous paragraph would prefer it. In this (exaggerated) example, however, perhaps it *would* be better that one person works double time, if that gives the rest a chance at 20% less work? For instance, those working 20% less might be willing to abstain from, say, 5% of their salary, and give this to the person working double time, and that person, in turn, might appreciate a 5 fold increase in salary as compensation for the unpleasant level of overtime. Another way to say this is that B might be a "Kaldor-Hicks improvement" over A. At SAS, however, there's no special premium or bonus to those who work the most each month, which partly makes this argument moot.

Both these two approaches were used in the TP system. A suitable value of 'N' from above was determined to be 1900. The term $u\left(TP - 1200\right)^2/100$ was used for each crew member in the cost function, where u is the discrete heaviside step function ($u\left(x\right)$ is 0 for all negative x, and 1 for all positive x.).

8.2.8 A further complication to fairness

Some crew members (or really, quite a lot of them) do not work full time. Several different work rates exist.

8.3 Experiments that are to be made:

1. Create a 'soft rule' for guarding against minimum rest time overrun, as per "An example cost function application: Minimum Rest Time", and evaluate the effects.

2. Implement the TP system, and evaluate its effects.

Chapter 9

Results

Many different schedules have been created, using different rule sets. The rule sets are:

Original This is the set of rules actually used in September 2005 by SAS Sweden, in operation.

Original+resttime This is basically the set of rules actually used by SAS in September 2005, but with the minimum rest time cost from section 8.1.

Phase 1 This rule set implements the first phase of the TP- idea.

Phase 2 This rule set implements the second phase of the TP-idea.

Most schedules have been created using the actual timetable for SAS Sweden during the month of September 2005. There can be significant differences in the time table from one month to another, so the results are really only valid for one particular month. A timetable for December was also created. While there are differences between September and December, the relative effect upon the schedule of the different rule sets remain similar. No rule set is month-specific. Typically, government rules change seldomly, and union agreements change once per year (approximately).

This section aims to present data, and without judgment evaluate it according to the Metric given in 7.2. See section 10 for a more subjective analysis of the data in this section.

9.1 Results from TP

The three rule sets “Original”, “Phase 1” and “Phase 2” are compared in order to determine the outcome of the TP system. They are evaluated according to the criteria in section 7.2.

9.1.1 Unmanned flights

For September some unmanned flights occurred in the original schedule. These were not due to under-capacity, but rather because illness among crew, reported just before schedule release. Since there was no time to re-run the optimization, the flights were not manned in the released schedule.

For December, some unmanned flights result. Some of the unmanned flights using the “phase 2” schedule were because a large number of long haul crew had vacation on the same day, and there were long haul flights a few days earlier, that they could not fly, because their guaranteed TP-free time could not fit before they were to go on vacation. This situation could probably have been avoided, if TP had been used during the vacation planning. One possible modification could be to let vacation count as rest time, when it comes to the guaranteed TP rest.

A graph has been created that shows the number of unmanned block hours, for each day in the month, see figure 9.1. Out of convenience, the same script has been used to generate both the September and December graphs, and erratically lists a non-existing 31th day of September.

You may wonder how it can be that some flight duty was scheduled unmanned in actual operations. The answer is that it is acceptable to leave some flights unmanned, since these can be covered by standby crew during operation. SAS always *operates* aircraft with at least the minimum legal crew complement.

Analysis: The Phase 2-rules might be a little less productive than the original. However, the difference is so slight as to be uncertain (the number of block hours *manned* is in the order of 10000 - so even 50 more or less is, in a way, just a small difference).

9.1.2 Fairness across individuals

In order to look at fairness between different individuals in the same category, we look at full time crew only.

A rather special graph has been developed. On the Y-axis, we have the number of TP earned. On the X-axis, we have all crew members, sorted according to how many TP they have. Thus, we have the plot of a monotonically increasing function, with a discrete domain. Such a graph is useful, since it allows us to quickly answer questions such as “How many TP do the 100 most TP-rich crew members have”? It also gives a good overview of fairness. If TP were completely fairly distributed - with each crew member receiving the same amount – the graph would be a straight horizontal line. Since the actual absolute number of crew isn’t what’s important, the X-axis has been graded in percent instead. The horizontal lines in the graph represent the average values of the different series.

From the graph, see figure 9.4, it can be seen that fairness is significantly increased for Phase 2, but virtually unchanged for Phase 1, compared to the Original.

9.1.3 Fairness across categories of crew

There are a few different categories of crew. The most common is regular full time crew. Then there are 75% and 80% part time employees. Then there are people with young children, who have a legal right for part time, and are guaranteed to come home every night (they cannot have layovers). Ideally, the 75% part time crew members should work exactly 75% as much as those with full time employment. TP is used as a measurement of “how much” crew work.

To evaluate this criteria, a plot similar to the one in the previous section has been created. However, in order to make comparisons between the categories easier to do, the ratio of actual TP to maximum allowed TP for each category is plotted instead of the absolute TP. See figure 9.5. Two different categories of crew are plotted, the 75% part time, and the 100% part time.

It can be seen from these graphs that fairness has been improved considerably in the Phase 2 runs. Those who work the most in the Phase 2 schedule work 15% less than those who work the most in the original schedule. Also, the graph slope is considerably less.

9.1.4 PBS bid satisfaction

There are some caveats to evaluating PBS satisfaction for a schedule.

Let us say we want to evaluate two schedules, A and B, for the same month and time table, but which have been generated with different rule sets.

Let us say that B grant almost no bids (because of a very restrictive rule set). If the bid ratio is used as measurement, it is conceivable that B will have a higher average bid ratio across crews. This is because both the reference roster and the production roster will have a low number of bids granted, but the bid *ratio* might be higher than for A.

Another possibility is to, say, use the reference roster of A as reference roster for both A and B. Now comparison is more meaningful. However, it can still be misleading. Let's say that A and B this time grant approximately an equal number of bids. Crew with few granted bids in A, but many in B, will now get an artificially high bid ratio.

A way around this problem, in turn, could be to use the maximum points value of the two reference rosters, as the reference points value. This would probably work pretty well.

Another approach is to simply use the number of raw points. This has the disadvantage that comparisons between crew of the same schedule are no longer meaningful. However, comparisons of the total number of raw bid points granted in different schedules, is meaningful. This approach has been chosen, because it is easy to implement.

A drawback with this approach is that it is difficult to scale the Y-axis of the graph. It has been chosen to clamp the bid-values at 1000 points. Crew which have been granted more points are considered no more happy than those who have only been granted 1000 points.

A graph similar to the previous graphs has been created, see figure 9.6. The plateau on 200 points is due to the fact that there are many rosters with only a single 200-point bid granted. From this it can be seen that, in September, the bid satisfaction was marginally better in Phase 1, than in the original, and worse in Phase 2. Interestingly, the situation is different for December. Here the bid satisfaction is marginally better in both Phase 1 and Phase 2, compared with the original schedule. Since the Phase 2 schedule is more flexible in some ways (crew don't need two full days off every time), and less flexible in others (it cares more about fairness), it is not really surprising that PBS satisfaction is better for some timetables (December) and worse for others (September).

The conclusion has to be that using the Phase 1 rules instead of the Original, does not affect the PBS satisfaction, whereas using the Phase 2 rules may sometimes have a small negative impact.

9.1.5 Proximity to minimum rest time rule

The Phase 1 and Original schedule have nearly identical rest time margins. See figure 9.7.

9.1.6 Proximity to the 270p and 90p rules

These two were plotted, but the plot is not reproduced in this document. The plots were identical for all rule sets.

9.1.7 Aircraft qualification problems

This plot also revealed absolutely no difference between different rule sets. The plot is not reproduced here.

9.1.8 Check-in/check out times

Only small differences to check-in times could be spotted. People generally start a little later in the day in Phase 2. However, the number of very early mornings did not differ appreciably between Phase 2 and the others. See figure 9.8.

9.1.9 Circadian rhythm

See figure 9.9. This plot shows the number of TP awarded to crew as a result of circadian rhythm disruptions. There is not much difference. The Phase 2 schedules do provide slightly less disruption to the circadian rhythm of crew. See section 10 for a discussion of why the difference is not greater.

9.1.10 Blank days

See figure 9.10. As can be seen from the graphs, the number of blank days using either Phase 2 rules is not worse (less) than the original schedule.

9.2 Results from minimum rest rule

The rule sets "Original" and "Original+resttime" are interesting to compare, in order to determine the effect of the Minimum Rest Time cost of 8.1. Let us now evaluate these three two sets according to the criteria in section 7.2.

There are no unmanned flights. This is expected, since all we've really done is that we've modified the cost of some schedules a little bit. The optimizer tries hard to avoid unmanned flights (because they are considered expensive), so without new rules, and without extreme cost functions, unmanned flights are unlikely to occur.

All of the graphs used for TP-evaluation were created, but for the "Original" and "Original+resttime" rule sets instead of the TP rule sets. However, all of them are not reproduced in this report, since most of them showed no visible difference at all between the two rule sets.

In fact, the only one showing a significant difference was "Proximity to minimum rest time rule". See figure 9.11 – the result here was no surprise. The margin to minimum rest time is greater, even if not by a big amount. Since the improved rest time cost-rule set only tries to increase rest time margin for flights that are frequently late, it is not obvious that there would be significantly larger rest time margin on average, but this does appear to be the case.

So now we have established that the new schedule is at least no worse than the original September schedule, at least according to our Metric.

Using real traffic data, we can examine on how many occasions people would actually overshoot minimum rest time.

Schedule	Nr Exceeded
Original September schedule	10
New September schedule	5

Since each occasion costs approximately 3000SEK, the amount saved in the new schedule is 15000SEK/month, or 180000SEK/year. This is really not a big deal.

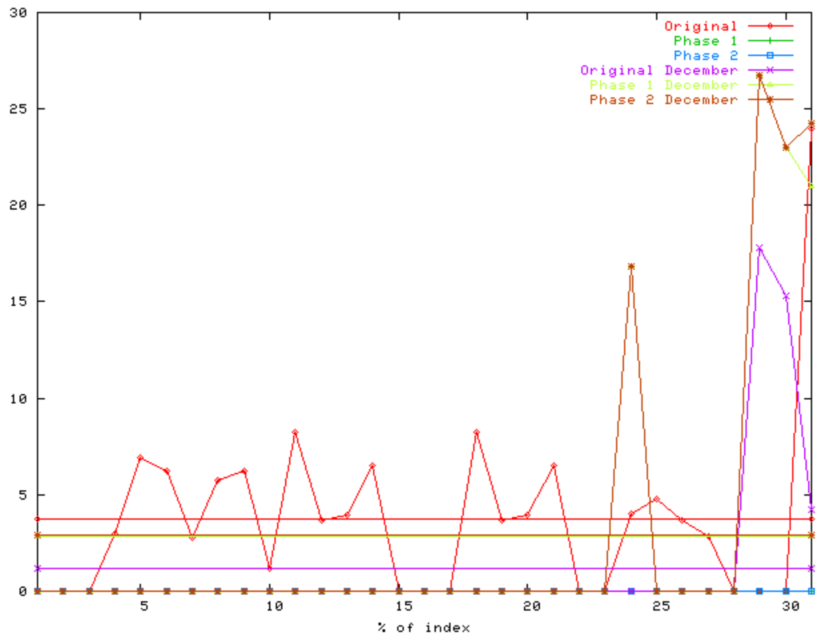


Figure 9.1: **Unmanned block hours** Number of block hours, each day of the month, that are left unmanned.

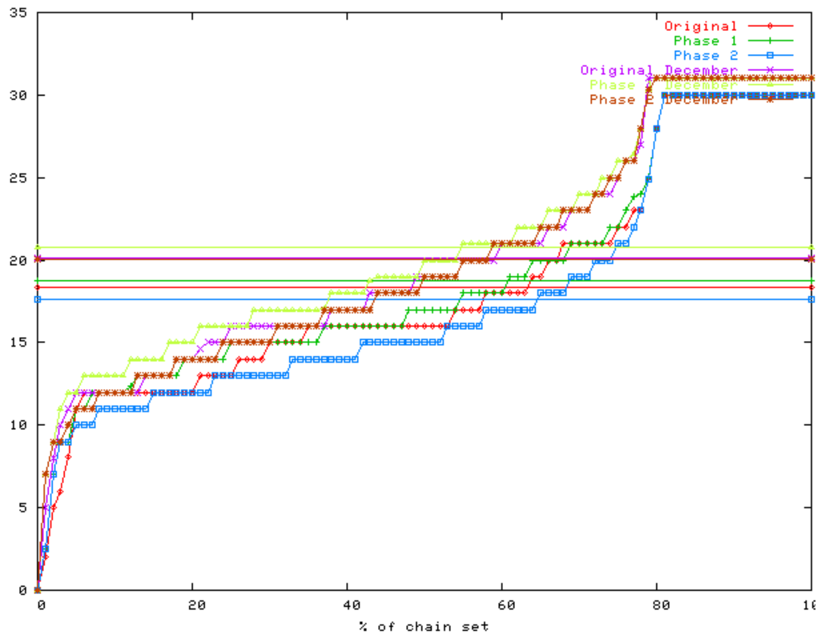


Figure 9.2: **Free days** Number of free days, all crew.

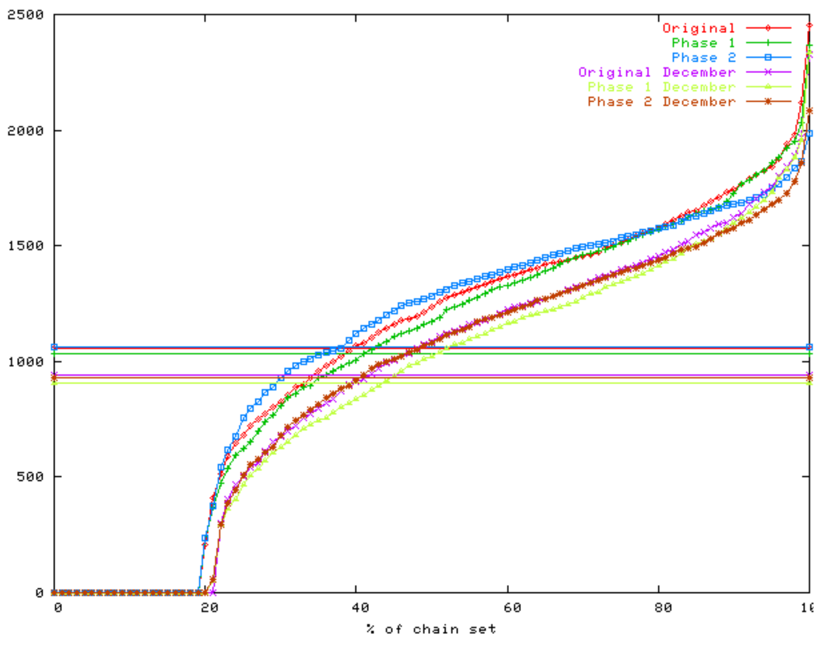


Figure 9.3: **TP, Total** TP, for all crew.

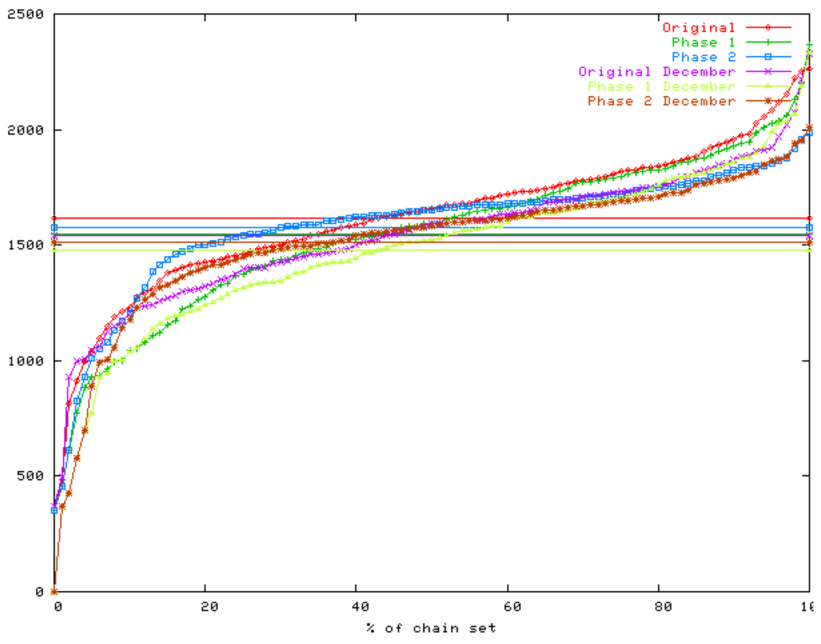


Figure 9.4: **TP, Fulltime** TP, Fulltime crew only.

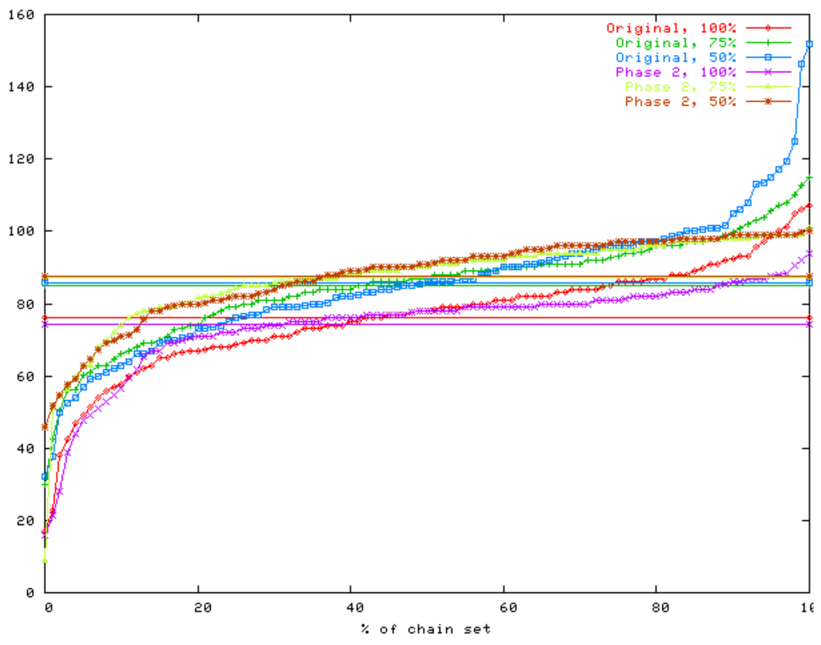


Figure 9.5: **TP Usage ratio** TP, the ratio of used TP to maximum allowed TP, for all crew. Expressed in percent.

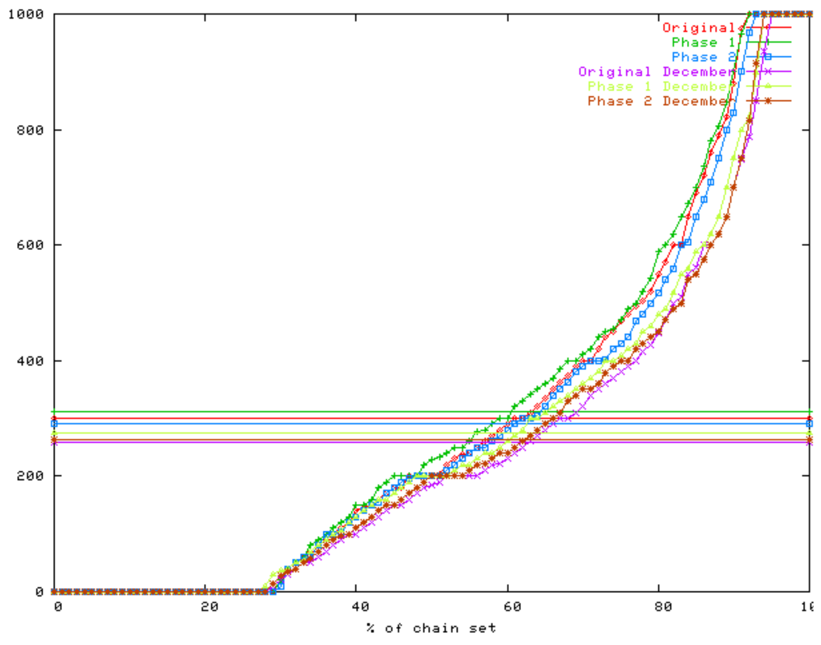


Figure 9.6: **PBS Bid satisfaction** All crew.

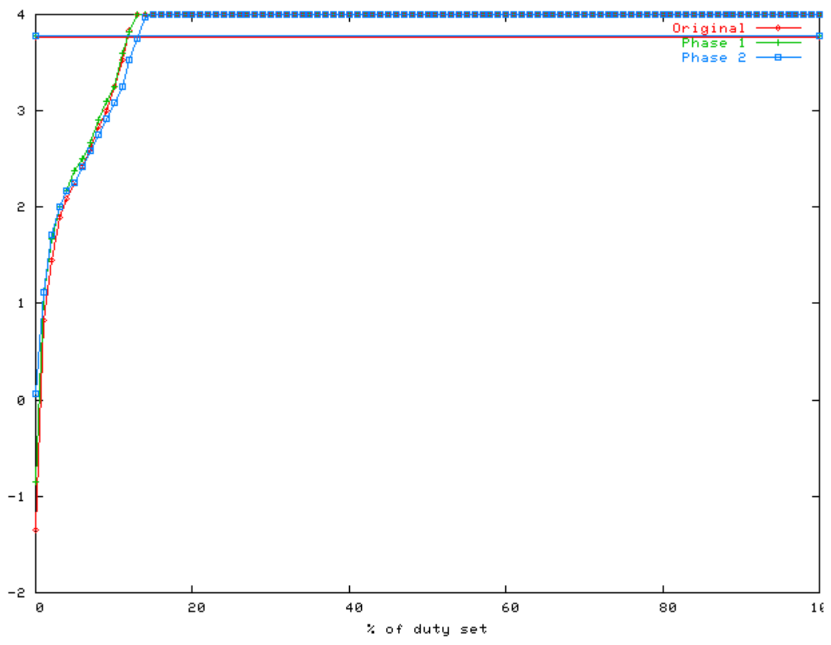


Figure 9.7: **Margin to resttime** Negative margin = too little resttime. Occasions with more than 4h resttime margin are included as if they had 4h margin.

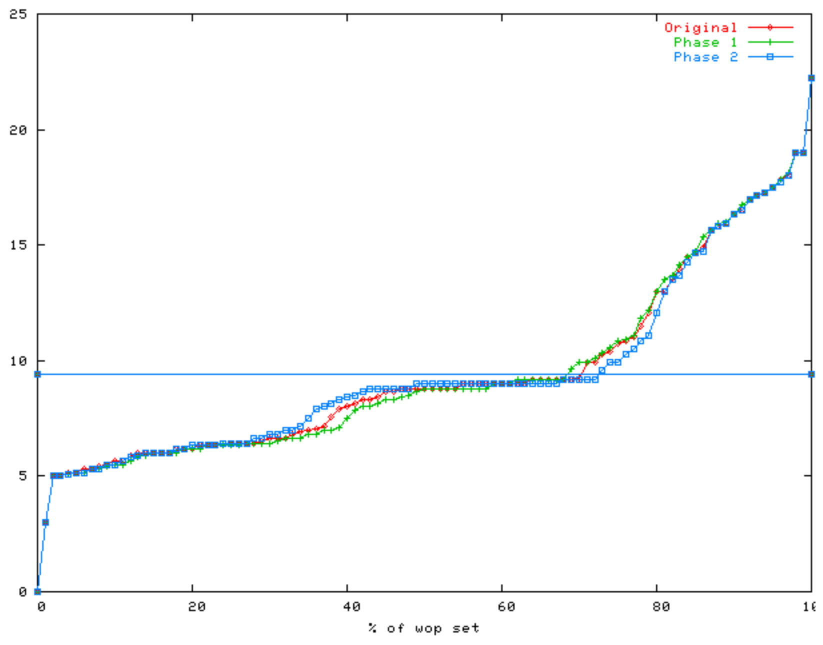


Figure 9.8: **Check-in** Check-in times for first day of working period.

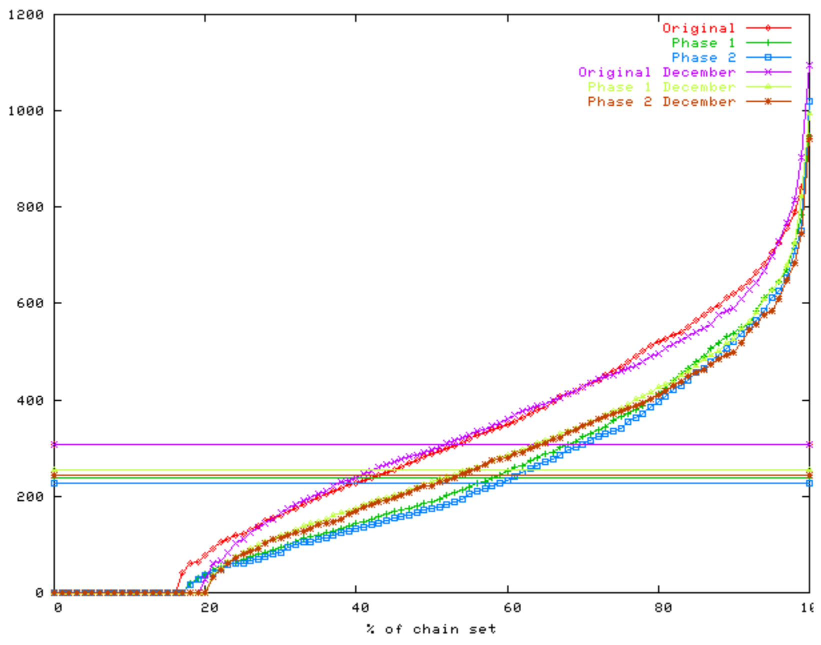


Figure 9.9: **Circadian rhythm** TP awarded to each crew for disturbing their circadian rhythms.

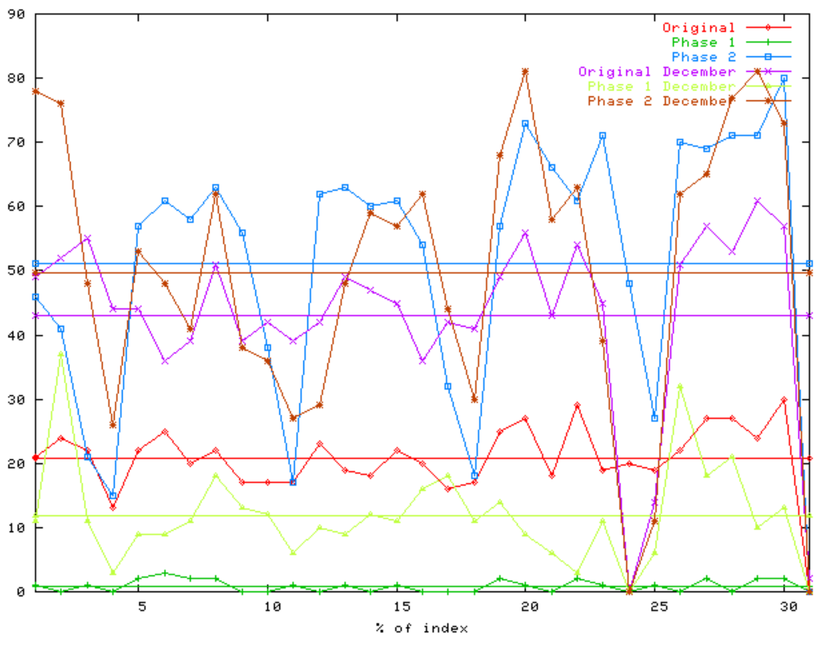


Figure 9.10: **Blank days** Number of blank days, each day of the month.

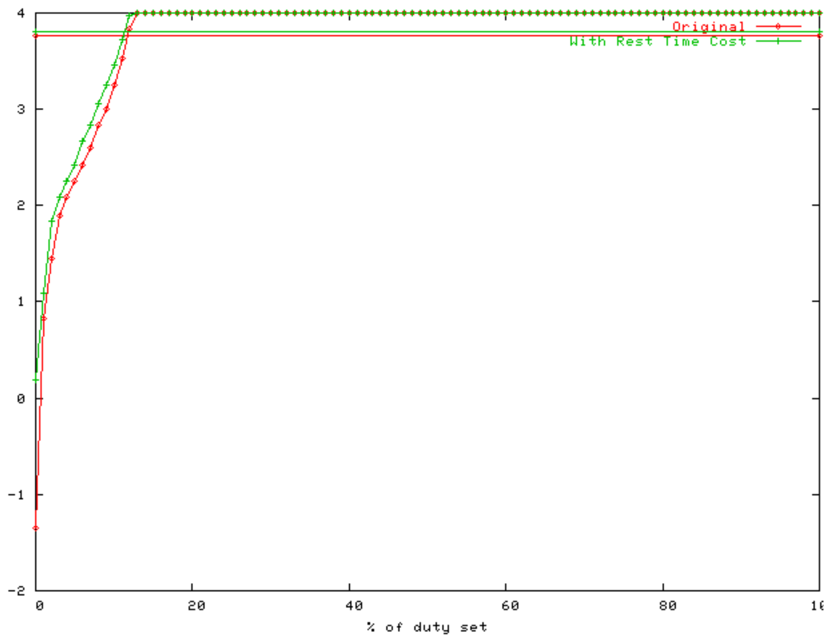


Figure 9.11: **Margin to resttime** Negative margin = too little resttime. Occasions with more than 4h resttime margin are included as if they had 4h margin.

Chapter 10

Conclusions

10.1 Overview

10.1.1 Minimum Rest Time

Using statistics from previous months, it proved possible to avoid half of the occasions where crew checked out so late that they had less time to sleep until check-in next morning, than the Union Agreement demands. Basically, there were no big surprises here. Some flights are more prone to being late than others, and by not scheduling any early morning flights the next day after one of these flights, it is intuitive that the amount of missed minimum rests should decrease.

However, this really was not much of a problem even with the old model, so the best possible improvement is not large.

10.1.2 TP

The TP model was used for scheduling, and the resulting schedules compared with the results from using the original model. Significant differences in the schedules produced were visible, and fairness was improved. Also, the TP system had the effect of minimizing the occurance of bad sleeping-patterns (violations of Circadian Rhythm).

From a practical perspective, a big problem with the TP system exists: It is not known wether or not it actually reflects reality. On approach towards determining this, would be to actually use it in operations, but this requires a new Union Agreement, and is a potentially costly experiment.

10.2 Fairness

A pressing issue today at SAS is that of fairness, both between individual crew members and between different groups of crew members. In particular, it has been noted that the full-timers work less than the name of their agreement seems to suggest. Using those with 75% part time duty as reference, you would expect those with 100% duty to work $100/75 = 133\%$ as much as those with 75% duty. However, the actual figure is more like 120% than 133%.

Now, if the part timers are to work less, somewhere, someone has to pick up the slack. Even though the full timers are not usually working as much as the union agreement allows, it is difficult to schedule more work for them. Partially since their work load might already be bordering on too high, in some cases, and partially since they are often scheduled to work as much as is at all possible according to the government rules (the BCL points).

All in all, there are signs that seem to suggest that real fairness across categories could be difficult to achieve. Correct relation of work between 50% and 75%, and between "fixed group" and "variable group" crew is probably achievable.

Also, fairness between individuals can probably be improved significantly.

10.3 PBS satisfaction

The lesson seems to be that PBS satisfaction is the first thing to take a hit when the optimizer gets into trouble trying to crew all flights. This is no surprise, of course, it is by design. The philosophy of the PBS system is that the preferences of crew should be honored, only in instances where several choices are equal in the eyes of the airline. The PBS system only affects the cost function, and by a relatively small amount. Everything that restricts the number of solutions possible for the optimizer will tend to hurt PBS bid satisfaction.

10.4 Circadian Rhythm

Those crew members with large circadian rhythm TP components in their schedules were studied. It turns out that all the worst disruptions are for long haul crew, where it is essentially unavoidable. Some disruptions, however, occur for crew on short haul flights. The latter could largely be avoided, if circadian rhythm was taken into account during pairing. Pairing is, as you recall, the act of putting together anonymous spells of work, that are assigned to individual crew members during Rostering.

One consequence of trying to avoid disruptions to circadian rhythms is that crew should get up at approximately the same time each morning, with a slow drift allowed over time. This means, that some crew would have to start early in the morning many mornings in a row. One argument for *not* trying to avoid circadian rhythm disruptions is that the crew themselves, according to experienced crew planners, generally prefer to not have many early mornings in a row. The idea being, that early mornings are unpleasant, and you need some time to recover after each and every one. The author of this paper believes that this argument may be a flawed one.

It might be that the reason why early mornings are considered unpleasant by crew is that there is at present nothing that prevents the days before the early morning from containing late afternoon duty. It is difficult to, say, go to sleep at 23:30 several nights in a row, then suddenly go to sleep 21:00, just because the next day is an early morning duty that starts 05:00. It could be that many crew members start their early shifts without having had a full night's sleep. Government rules guarantee that a crew member must have had access to suitable sleeping arrangements, for at least 8 hours, before each working day. However, there's no requirement that the crew member must actually fall asleep, or be tired enough to have a chance of falling a sleep. Perhaps, in the real world, it is of great importance whether you actually did fall asleep or not.

It is interesting to note that there seems to be a strong consensus in the medical community that irregular sleep is harmful to the body. However, there seems to be less consensus as to how well humans can adapt to working in the night, and sleeping in the day.

10.5 Blank days

One thing that was noted during the work of implementing the TP system in rave and CAS, was that the number of blank days assigned for a certain month could vary greatly with the cost function used. This makes it more difficult to compare schedules with each other, than would otherwise have been the case. With any scheduling philosophy (TP, the original one, or others), it is likely that better schedules can often be had at the expense of a reduced number of blank days.

It can also be noted that blank days do not fit very well into the TP system. Blank days are always one whole day, while time off after working periods is awarded on a per-hour basis.

10.6 Summary

TP (as described in this document) represents a feasible scheduling system, and it does distribute work load more evenly, both across crew and across time for each individual crew member. The disadvantages are chiefly in PBS fulfillment, and possibly a slightly lowered productivity in some cases. Also, by modeling the factors that contribute to crew overshooting the minimum rest time rule more closely, the occurence of missed minimum rest time can be reduced by half.

10.7 Considering Circadian Rhythms in Pairing

It would have been interesting to try and further minimize disruptions to the circadian rhythms of crew, by considering sleep patterns in pairing.

This would probably be rather easy to do, the main factor that has prevented this from being done in this work is simply that the author is not familiar with the Pairing-tool, PAC.

In fact, one may try to introduce all of the TP concept into pairing.

Chapter 11

Ideas for Further Research

11.1 Tune the TP model

A researcher with a background in medical science, behavioural science or psychology, could tune the TP model so that its parameters actually reflect the effect of different schedules on the human body and pscyhe.

11.2 Remove artificial costs

It could be interesting to try and replace all artificial costs with actual costs. This will make comparison of different schedules much easier. Perhaps artificial costs are needed to guide the optimizer. Whether or not this has to be done, is something that needs research. Also, research should try to find what kinds of trouble occur when striving to remove artificial costs. Perhaps it is difficult to model complex costs, that are not decomposable into one part per employee (see the ’count things twice’-problem, 7.4)?

11.3 Fuzzy Logic

It might be interesting to adopt a fuzzy-logic view of scheduling. For example, we might say “No one should have to start early in the morning the first few days after vacation”. The question: “What is an early morning, exactly” then becomes apparent. Also, what does “a few days” mean? One strength of fuzzy logic is that it is well suited to dealing with this kind of classification problem. A certain duty might be 85% early morning, and 23% ’a few days after vacation’. Challenges:

- Implementing (in Rave) fuzzy AND, OR and perhaps other logical elements.
- Implementing (in Rave) a suitable defuzzification algorithm.

Using Fuzzy logic with PBS could solve the problem that now occurs when a crew member only places one bid. Since the optimizer then has to choose between granting 100% or 0% of the person’s bids, and since it’s more likely for it to give a person 100% than 0%, placing only one bid can be used as a means to get higher priority by the optimizer. Using fuzzy logic, however, a bid could be 50% granted, and the problem goes away. Care must be taken, however, if this kind of system is to be appreciated by crew. A person who wants 24h off during a certain day, in order to be able to attend e.g a wedding, might not be happy at all if, for instance, one hour in the middle of that day is scheduled with duty, and the system says his bid is 96% granted!

11.4 Using Neural Networks to classify WOPs

It would be interesting to try and train a Neural Network to determine how burdensome a particular Working Period is for crew.

A way to do this might be to first present a set of working periods, perhaps 50, to a reference group of crew, and let them rate the working periods on a scale from, say, 1 to 5. This set would then be used as training set for the Neural Net.

A couple of features would be identified for each working period, perhaps things like “contains charter flights”, “more than 3 days”, “contains early mornings”, “contains >10h working days”, etc.

Then a neural net would be trained to correctly classify the working periods in the training set, based upon the features present in the working sets.

Experiments could be made to determine if analogue or digital inputs work best, and what resolution in classification can reasonably be achieved.

Some sort of code generator would probably be in order to actually implement the neural network in the very restrictive Rave-language. It should still be within the capabilities of the rave language, however.

The output of the Neural Net would then be used to distribute pairings in a fair way among crew. Gaining acceptance in the Union for letting a Neural Net govern the work load of crew is probably a challenging task.

11.5 Understanding Emergent Behavior

It would be very valuable if some added insight could be made into emergent phenomena in scheduling and rostering. At the moment, rule-changes do not always yield the expected result. Therefore, whenever a change of parameters, rules or cost function is proposed, an entire optimization run, or several entire optimization runs, have to be carried out to evaluate its effects. It would be nice if some certain key performance indicators could be calculated without requiring a complete optimization solution to be created. This is probably an incredibly difficult and potentially unrewarding task.

Chapter 12

Dictionary

This section is intended as a reference for some of the special words used in this text.

Airhostess/host Person working on board the aircraft, serving meals and generally providing service to the passengers. Also responsible for aiding evacuation during an emergency.

Base The office at which flying personell are officially employed. Trips start and end at a base. The Swedish branch of SAS has only one base, Stockholm/Arlanda.

Blocktime Time from push-back to arrival at destination airport gate.

Cabin Crew Airhosts/hostesses, Stewards and Pursers.

Chain = A sequence of flights to be flown by one crew member, during a planning period. The planning periods at SAS are one calendar month, plus the first week of the following calendar month. This is the period of time considered by the solver for one solving operation.

Construction Initial construction, the first generation-step.

Deadhead Crew flying as passengers on a plane.

Flight = In this work, the same as ’leg’.

Flight Deck = Pilots + flight engineer (for those rare planes that require one)

Generation Generation step, creating chains for the IP optimizer to choose between.

Hard rule A rule that may not be broken under any circumstances.

IP optimization Refers to the process of selecting chains out of the generated chains.

IP Optimizer The component of the carmen software doing IP Optimization.

Leg A take off, followed by a landing.

LH Long Haul. See 2.

Purser Leader of the cabin crew team on each flight.

Roster The generated schedule for one crew member.

Schedulability How easy a crew member is to schedule on duty. All else being equal, a 100% employee is more schedulable than a 25% part-time employee. Some employees have medical limitations that preclude them from certain flights - this does also reduce their schedulability.

SH Short Haul. See 2.

Slot = In Crew planning: one task on a specific leg. Each leg will have one slot for the pilot, one for the copilot, and one for the purser. Depending on what aircraft type is flying the leg, there'll be a different number of air hosts/air hostesses.

Soft rule A rule which may be broken to some degree, at a cost.

Solving Refers to solving the entire rostering problem.

Steward Airhostesses/hosts can be promoted to Stewards. These have more training as chefs.

Chapter 13

Acknowledgements

Thanks to Anders winberg at SAS for patiently and competently answering all my questions about crew planning and scheduling. Thanks to Christian Schulte at KTH for being my examiner. Finally, thanks go to Hans Norman (also at SAS) for allowing me to use his beautiful aircraft photos.

Bibliography

[1] A. Caprara, F. Focacci, E. Lamma, P. Mello, M. Milano, P. Toth, and D. Vigo. Integrating constraint logic programming and operations research techniques for the crew rostering problem. *Software Practice and Experience*, 28(1):49–76, 1998. URL `citeseer.ist.psu.edu/caprara98integrating.html`.

[2] Carmen Systems. *Advanced APC for Pairing*.

[3] Rigas Doganis. *The Airline business in the 21st century*. Routledge, 2001. ISBN 0415208823.

[4] Claude P. Medard et al. Airline crew scheduling - from planning to operations. Technical report, Carmen Systems AB, 2005. URL `http://www.carmen.se/research_development/research_reports.htm`.

[5] Niklas Kohl et al. Airline crew rostering: Problem types, modeling and optimization. Technical report, Carmen Systems AB, 2004. URL `http://www.carmen.se/research_development/research_reports.htm`.

[6] Ranga Anbil et al. Column generation and the airline crew pairing problem. *Documenta Mathematica – Journal der Deutschen MathematikerVereinigung*, proceedings of the ICM(III), 1998.

[7] Randolph W. Hall. *Handbook of Transportation Science, Second Edition*. Kluwer Academic Publishers, 2002.

[8] Frederick S. Hiller and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 2001. ISBN 0072321695.

[9] Curt Hjorring. Solving larger crew pairing problems. Technical report, Carmen Systems AB, 2004. URL `http://www.carmen.se/research_development/research_reports.htm`.

[10] Dominic Hyde. Sorites paradox, 2005. URL `http://plato.stanford.edu/entries/sorites-paradox/`.

[11] SAS och kabinforeningarna. Kollektivavtal.

[12] Virginia Postrel. Operation everything. *The Boston Globe*, 2004. URL `http://www.boston.com/news/globe/reprints/062704_postrel/`.

Appendix A

Work Carried Out

A.1 Rave code

Ca 500 lines of rave code, in two files, has been written. Most of this is TP modeling. A smaller mass of code is for testing and evaluation.

A.2 Python code

A total of approximately 800 lines of python code has been written, spread over 5 files. The python version used is 2.2.

A.2.1 Introducing actual traffic data into CAS

A python script has been written that converts traffic data (information of flights, delays, crews etc, in actual operation) into a format readable by Rave.

A.2.2 Correlation of delays

A python script that finds the coefficient of correlation between delays of specific flight numbers in different months was written.

A.2.3 Automatic extraction of graphs from CAS

A python script that interfaces with CAS, programatically loads the results of different optimization runs, and then interfaces with gnuplot to create graphs, was written. The data that is graphed is determined by an XML-file, that the user can edit to graph different quantities. The XML-file is read through use of the XML-parser build into Python.

A.3 Carmen publisher-reports

One new report in the Carmen Report Language has been written. This report gathers information about TP, such as standard deviation and maximum value, and other parameters.

One of the existing reports, Assignment_statistics_CA, has been marginally extended with information about TP.

Information about TP has been added to the data-windows in the CAS GUI. Holding the mouse over an activity shows how many TP this activity yields.