Annals of Operations Research 127, 223–257, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Airline Crew Rostering: Problem Types, Modeling, and Optimization

NIKLAS KOHL niklas.kohl@carmenconsulting.com
Carmen Consulting, Købmagergade 53, DK-1150 Copenhagen, Denmark

STEFAN E. KARISCH stefan.karisch@carmensystems.com
Carmen Systems Ltd., 1800 McGill College Avenue, Suite 2800, Montreal, Quebec, Canada H3A 3J6

Abstract. Airline crew rostering is an important part of airline operations and an interesting problem for the application of operations research. The objective is to assign anonymous crew pairings either to personalized rosters or to anonymous bidlines which subsequently will be assigned to individual crew members. Compared to the crew pairing problem, crew rostering has received much less attention in the academic literature and the models presented have been rather simplified. The contribution of this paper is two-fold. First, we want to give a more comprehensive description of real-world airline crew rostering problems and the mathematical models used to capture the various constraints and objectives found in the airline industry. As this has not been attempted in previous research, we think it serves a purpose to reveal the complexity of real-world crew rostering to readers without industrial knowledge of the problem. Second, we want to present the solution methods employed in a commercial crew rostering system, in whose development we both have been involved. The Carmen Crew Rostering system is currently in use at several major European airlines including British Airways, KLM, Iberia, Alitalia, and Scandinavian Airlines (SAS) as well as at one of the world's largest passenger transportation company Deutsche Bahn (German State Railways). During the development of the Carmen Crew Rostering system, we have gained valuable experience about practical problem solving and we think the system constitutes an interesting case in the application of operations research.

Keywords: crew rostering, crew scheduling, airline applications

1. Introduction

1.1. Background

An airline receives revenue from passenger tickets and pays for aircraft, fuel, crew and airport usage. After costs for fuel, crew costs constitute the second largest expenses of an airline. Since profit is the difference between revenue and cost, cost efficient crew planning is of major importance for airlines. A cost reduction of a few percent usually results in annual savings of tens of millions US dollars for large airlines. Due to the potential for significant cost savings, operations research techniques were applied in the area of crew scheduling already at an early stage.

Due to its complexity, crew planning at airlines is usually divided into a *crew pairing* and a *crew rostering* (or *assignment*) phase. Firstly, anonymous pairings (or crew rotations) are formed out of the *flight legs* (flights without stop over) such that the crew needs on each flight are covered. So a *pairing* is a sequence of legs to be assigned to one or more crewmember working in one or more crew positions (ranks). The crew positions and the number of crewmembers a pairing must be assigned to is referred to as *crew complement.* Then in crew rostering, the pairings together with possible other *activities* such as ground duties, reserve duties and off-duty blocks are sequenced to rosters and assigned to individual crewmembers. Rostering is usually done one month at a time and each roster usually has some history. In both problems, complex rules and regulations coming from legislation and contractual agreements have to be met by the solutions and some objective function has to be optimized.

Most of the published work relates to crew pairing, see, e.g. (Anbil et al., 1991, 1992; Andersson et al., 1998; Barnhart and Shenoir, 1996; Barnhart, Hatay, and Johnson, 1995; Barnhart et al., 1994; Barutt and Hull, 1990; Desaulniers et al., 1997; Gershkoff, 1989; Graves et al., 1993; Hjorring and Hansen, 1999; Housos and Elmroth, 1997; Lavoie, Minoux, and Odier, 1998; Vance et al., 1997). The reason being that most of the cost benefits can be achieved by having productive pairings that minimize costs. In crew rostering, the minimization of costs is also important, however, quality of life aspects for crew are considered as well. The objective is a combination of cost efficiency and the creation of rosters that satisfy crew. In this paper we focus on the crew rostering problem, i.e., the creation of rosters for individual crewmembers, which usually takes place 2–6 weeks before the flights are operated.

The standard methods, presented in the literature and applied in commercial crew rostering systems, are based on the "generate-and-optimize principle." In the master (optimization) problem, a set partitioning type problem is solved to select exactly one roster for each crewmember such that the demands of the activities are met, the solution satisfies constraints between several crewmembers, and the objective is optimized. Since it is not possible to have an explicit representation of all possible rosters, the master problem is always defined on a subset of all possible rosters.

In the (generation) subproblem, a large number of legal rosters is generated. This can be done by partial enumeration based on propagation and pruning techniques as employed by, e.g., the Carmen Crew Rostering system and as described in this paper. An alternative is to solve a constrained shortest path problem where the constraints ensure that only legal rosters are generated, and where the objective function is equivalent to the reduced costs of the roster with respect to the solution of the continuous relaxation of the master problem defined on the previously generated rosters (Gamache et al., 1998). The latter approach is known as constrained shortest path column generation. Here the subproblem is solved to optimality and one can prove that it is possible to obtain the optimal solution to the entire problem without explicit enumeration of all possible rosters. In either case one can iterate between the subproblem and the master problem.

1.2. Problem types

Even though crew pairing problems are different from airline to airline with respect to rules and costs, the main characteristics remain the same. The rule structures are comparable and objective functions are based on pay and reflect mainly real costs, as mentioned above.

In contrast to pairing, crew rostering can be done in various ways following different approaches. In North America for instance, at most airlines the rostering is done in two steps: first, anonymous rosters (or so-called “bidlines”) are created which are then assigned to individuals based on bids for these anonymous rosters. We refer to this rostering approach as the *bidlines approach.*

At most European airlines, however, individual rosters are constructed directly for each crewmember, which we refer to as *personalized rostering.* This approach can be based on fair share, i.e., all crewmembers should have rosters satisfying certain quality criteria, or on individual preferences. In the latter case, crewmembers express personal preferences that are considered during the creation of the individual rosters. These preferences can be awarded according to seniority, i.e., the most senior crewmembers get a maximum of their preferences granted limiting the awards of less senior crewmembers, or again on a fair share basis.

When comparing the different rostering approaches, benefits of the bidlines rostering process are on the crew side. By bidding for a specific line, a crewmember knows exactly what he/she will get if the bid is granted. This is in contrast to *preferential bidding* (personalized rostering with preferences) where crew “only” express preferences for certain attributes of their rosters without knowing exactly how the roster will look like. However, there are preferential bidding systems on the market that provide immediate feedback for crewmembers during the bidding phase and determine important characteristics of the expected rosters a-priori. Drawbacks of bidlines are greater costs that occur when the bidlines cannot be assigned entirely to individuals due to conflicts with pre-assignments and vacation days, and some pairings of the bidline can hence not be assigned. These conflicts are often bid for by crew since the pay is determined by the original roster and not by what is actually assigned. Moreover, the planning process consists of more steps than in personalized rostering, where one individual roster for each crewmember is generated once and then published.

From a solution point of view, however, the different rostering approaches are quite similar, and from a modeling point of view, they differ mostly in the formulation of the objective function. As mentioned above, the goal is to minimize costs while considering quality of life criteria. Since these different rostering principles are applied differently at airlines, and are usually combined and extended, there is a demand for flexible and generic rostering systems, that support both the modeling of the different rostering environments and provide proper optimization methods to solve the resulting problems effectively and efficiently.

1.3. Literature survey

We now provide an overview of the scientific literature in the area of airline crew rostering.

Rostering with the bidlines approach has been described in (Campbell, Durfee, and Hines, 1997; Christou et al., 1999; Jarrah and Diamond, 1997). The first two of these references describe applications at FedEx and Delta Air Lines, respectively. The problem is solved using meta-heuristics. In (Jarrah and Diamond, 1997), a semi-automatic system is developed based on column generation, where the user influences which subset of columns is generated.

For personalized rostering, the set partitioning model, as described in this paper, has been mostly used in the literature. The GERAD research center developed a column generation approach where the subproblems, generating rosters for individuals, are solved as constrained shortest paths. The approach is exploited in the commercial AD OPT ALTITUDE system (AD OPT Technologies, 2003) and described with several applications, e.g., to the preferential bidding problem at Air Canada, in (Gamache and Soumis, 1998; Gamache et al., 1998, 1999). In (Lasry et al., 2000), the use of ALTITUDE for crew planning at Air Transat is described.

In (Day and Ryan, 1997) the rostering of cabin crew for Air New Zealand's short-haul operations is described. The problem is decomposed into the problem of assigning off days followed by the problem of assigning the pairings and other activities. Both these problems are solved by column generation, but the total number of columns in each of the two problems is rather limited compared to the case where off day assignment and pairing assignment is optimized simultaneously. The approach works well because the majority of pairings are one day pairings, thus the off day assignment does not cost much in terms of flexibility in the pairing assignment.

An exact IP approach that exploits some valid inequalities is presented in (Cappanera and Gallo, 2001) and applied to solve small instances from a medium-sized Italian carrier. (El Moudani et al., 2001) suggests a heuristic approach to solve a bi-criteria version of the rostering problem and apply it to medium-sized problems – the two criteria considered in the objective function concern costs of the rosters and crew satisfaction.

Within the Parrot project (Parrot, 1997) a solution approach combining column generation with constraint programming (CP) has been developed. Constraint programming is used to prune the search for rosters whereas the master problem, selecting roster for all crewmembers, is solved as a linear program. The work is documented in (Fahle et al., 2002; Kohl and Karisch, 2000; Sellmann et al., 2002) and has been applied to instances of a medium and large size European airline.

The papers (Dawid, König, and Strauss, 2001; König and Strauss, 2000a, 2000b) look at different modeling aspects of crew rostering and describe an implicit enumeration heuristic with propagation techniques. The heuristic has been implemented in the SWIFTROSTER algorithm and applied to data from a medium-sized European airline.

Article (Freling, Lentink, and Wagelmans, 2001) presents a decision support system for airline and railway crew planning based on a branch-and-price solver, a system that is marketed and sold by ORTEC (ORTEC, 2003). The paper describes an example for a small Dutch charter airline and refers to examples for Dutch Railways presented in (Freling, Lentink, and Odijk, 2001).

Various aspects of the Carmen Crew Rostering system have previously been described in (Hjorring, Karisch, and Kohl, 1999; Kohl, 1999, 2000). A related aspect, the partial evaluation of rules and restrictions and its application within the Carmen framework, has been described in (Augustsson, 1997).

Related work for the railway industry is also of interest, since the crew rostering problems are relatively similar. Heuristic algorithms employing OR, CP, and hybrid approaches are presented and applied to rostering problems arising at Italian Railways in (Caprara et al., 1998a, 1998b, 1998c). The work describes the modeling of the problem and effective solution methods.

1.4. Outline

We conclude this section with an outline of the remainder of this paper. Section 2 describes the problem in more detail, thereby focusing on typical rules and regulations and on different objective functions used in crew rostering. The following section 3 provides a mathematical model of the crew rostering problem. Initially we formulate the basic rostering problem as a set partitioning problem. Gradually this model is extended with generalized and additional constraints. Together these two sections constitute the part of the paper which describes the real-world crew rostering problem.

The following two sections are devoted to the description of a commercial crew rostering system that we have participated in the development of, namely, the Carmen Crew Rostering system. Section 4 discusses the different solution methods (algorithms) employed in the system and in section 5 we summarize some of the practical experiences we have had with the system. We conclude this article with a summary and an outlook of future development.

2. Problem description

2.1. Problem representation

In this section, we give a detailed description of the most important aspects of the airline crew rostering problem and elaborate on the main ingredients. Figure 1 provides a graphical representation of crew rostering catered to the different problem types described above. In the following we describe the figure and point out possible exceptions.

The input for a crew rostering problem consists in general of crew information, activities to be rostered, rules and regulations, and objectives for the creation of the rosters. However, when creating bidlines, i.e., anonymous rosters, individual crew aspects are not considered.

When producing personalized rosters, each crewmember's personal records, qualifications, pre-assigned activities, and vacation days are given. The records usually contain accumulated attributes such as hours flown during the current calendar year. Other

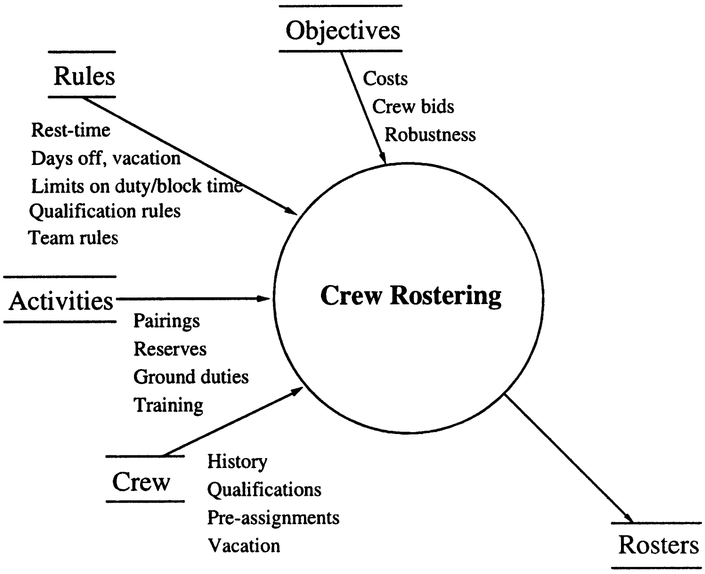

Figure 1. Representation of the airline crew rostering problem.

values of interest are due dates for training or possible exceptions from certain rules and regulations. Personal qualifications contain for instance information about the equipment the crewmember can operate or a list of destinations the crewmember cannot fly to. For cabin crew, language proficiency is an important qualification for international flights. Pre-assigned activities could be training, office duties or medical checks.

The set of activities which are to be assigned consists of pairings, reserves (e.g., airport and home standby duties), ground duties (e.g., medical checks), and training activities (e.g., simulator training and courses). In the bidlines approach, only pairings and reserve blocks are usually considered as input. In the following, we will refer to the activities as tasks when they are assigned to an individual.

The rules and regulations and objective functions are described in detail in the following subsections.

2.2. *Typical rules and regulations*

The rules and regulations express the conditions under which a roster is considered legal. Some of these may be due to legislation and governmental agencies (e.g., the Federal Aviation Administration in the USA), some may be imposed by the airline itself, and others are due to agreements between the company and employee unions.

In this subsection, we focus mainly on a detailed description of typical rules and regulations which are applied at airlines. The different possibilities of how rules are modeled and implemented are discussed in section 3. In the description of the rules, we distinguish between so called *horizontal, vertical* and *artificial rules.* This classification corresponds to the Gantt-charts presentation of rosters and solutions in the Carmen Crew

Rostering system. Rosters are represented as rows and hence rules which only depend on one roster and crewmember are called horizontal. Vertical rules combine information across several rows, i.e., several rosters and crewmembers. Artificial rules can be of both types and represent additional constraints which are used to exclude (feasible) solutions of poor quality and to better direct solution methods.

2.2.1. Horizontal rules

These rules apply to single rosters only and do not consider any other rosters in the solution. The horizontal rules are concerned with attributes of the crewmember for which the roster is generated and with properties of the assigned activities. The majority of rules of crew rostering problems belong to this category. Large European airlines for instance usually have more than 100 horizontal rules and regulations to consider. It is not possible to give a comprehensive overview of all rules that may exist at an airline. Instead, we present what we consider important types of rules and give some concrete examples.

(Person, task, time)-compatibility. These rules are "easy" in the sense that one can calculate (as preprocessing to the scheduling) the legality of any (person, task, time)-combination. The incompatibility may be due to lack of qualifications or need (e.g., medical checks) or pre-assignments (including planned absences). A special, but important, case occurs when a crew member changes function during the planning period. Before the day of the function change the crewmember must be assigned in say, cabin attendant position, but after the function change the crew member must be assigned in the purser position. More complex rules of this type may also exist. For example, the (person, task, time)-compatibility may depend on the assignment of a previous task. This type of rule does obviously not occur in the first phase of the bidlines approach, i.e., the creation of the anonymous rosters.

Rest time between tasks. These rules are also generally easy since they determine whether two tasks can follow immediately after each other. The calculation of the rest time itself may be rather complex since it may be influenced by many factors such as the length (days, duty time, block time, etc.) of the previous pairing or the last duty period of the previous duty period (possible night duty, time zones crossed, etc.).

In many cases, attributes of the next task or the first leg of the next task may also influence the rest time. Typical attributes of this kind are the type of task (flight, ground or reserve duty) and the time of the day the task under consideration starts and/or ends.

In some situations the rest time can depend on more than the previous and next task. For example, it can be related to other previous tasks or to the person. In the latter case, medical restrictions may require different rest times.

Rest day patterns. Pairings and other tasks are naturally grouped into working periods. In short and medium haul a working period typically consists of several tasks but in long haul a working period is often a single pairing. The duration of a working period, usually measured in working days, is always limited. In short and medium haul the limit is usually between 4 and 8 days, in long haul it is usually slightly more than the length of the longest legal pairing length. At the end of a working period some (often two) calendar days without any duty must be given.

Often, the rest time produced by the last task of the working period is not included in this off-duty period and the required rest may be extended if the working period is particularly long. Moreover, certain patterns might need to be satisfied in a longer time period, e.g., at least four consecutive days off in a calendar month.

Some airlines operate “fixed” rest day patterns. The crewmembers always work for a period of a days and are afterwards off-duty for b days.

Accumulated values. All airlines have rules limiting the number of block hours, often on a monthly and yearly basis. Also there are rules guaranteeing a minimum number of days off. Many other accumulated values may also define rules. Examples include: duty hours and number of “heavy” duties (e.g., many flight legs, many block hours or many duty hours).

In most cases, the accumulated value is only bounded from one side, but exceptions occur. For example, Gamache et al. (1998) report that the number of monthly block hours at Air Canada must be in the interval from 70 to 78.

The constraint on the accumulated value may be imposed in fixed intervals, such as calendar month or calendar week, or in gliding intervals, such as all 7 consecutive calendar days or all 168 consecutive hour periods. Also it may be imposed on a working period regardless of its length.

Several constraints on accumulated values are defined on a time period longer than the usual planning period. Examples include limits on block hours and days off defined on 3-month periods and the calendar year. Also the yearly vacation belongs to this category. It must be given at some time, but not necessarily during any particular planning period. Another example could be highly attractive tasks where each crewmember may be guaranteed at least one of these per calendar year.

Further example of horizontal rules. We give one more example of horizontal rules that reflects the complexity of the rules and regulations.

Traveling across several time zones is tiring and some time for acclimatization may be needed. In long haul crew rostering, this often means that an eastbound pairing satisfying some criteria on, e.g., duration and time zones crossed cannot, during the next a days, be followed by a westbound pairing satisfying some, possibly other, criteria on, e.g., duration and time zones crossed. Another implementation of the same principle is to let the minimum number of days off between two pairings depend on the maximal difference in time zones on the two pairings. A third implementation is to grant an extra day off between the pairings (provided that some criteria are satisfied) unless it is the first time this occurs during the month.

2.2.2. Vertical rules

Vertical rules concern more than just one roster. In most cases, they depend on a subset of rosters, but there are also some that concern the whole schedule. The basic building blocks of vertical rules are:

1. Crew complement.
2. Qualification-type constraints:
    - Task-qualifications;
    - Leg-qualifications.
3. Global constraints.

Crew complement. Even though the crew rostering problem is usually decomposed by crew, different activities require different crew complements. For example, for a short-haul cockpit crew problem, flight pairings are usually assigned to one captain and one first officer. However, certain ground duties such as simulator training could require two captains and one first officer. Moreover, certain tasks may require additional positions such as instructors. Even though crew complement can be seen as a special task-qualification constraint (see below), we treat it separately since it is the most important type of vertical constraint.

Qualification-type constraints. Particular tasks may require that some crewmembers hold some particular qualifications. This is mainly a cabin problem, but also occurs for cockpit crew. The constraint itself can be defined on the task, or the flight leg level, i.e., the constraint applies to a task, or a flight leg. For example, if all crewmembers on a flight leg fly the same pairing, or the qualification requirements are defined for the pairing only, it is more efficient to deal with it on the task level. The following lists some important examples of qualification-type constraints and state on which level they are defined.

- *Inexperienced crewmembers*. The number of inexperienced crewmembers on a particular pairing is usually bounded. In cockpit crew problems, at most one inexperienced pilot can operate a pairing. This constraint is usually found on the task level.
- *Must fly together*. In many cases some crewmembers must be assigned the same activities, where the set of activities is not predefined. This rule could apply for instance to married couples for a certain number of pairings per month.
- *Incompatibilities*. Often, some crewmembers are incompatible in the sense that they cannot be assigned to the same pairing – and possibly not to some ground duties as well – due to conflicting personalities. This is primarily a cockpit problem and it occurs most frequently in long haul where the crew has to cooperate for a quite long period of time. This constraint is typically an example of a task-qualification.
- *Language qualification*. In the following, we give an important example of a typical leg-qualification constraint. However, note that most of the task-qualifications listed above could appear on a strict subset of the legs of the pairing and would then be leg-qualification constraints. In long-haul cabin crew problems these constraints are usually present. For example, on a Copenhagen–Bangkok flight leg we may need five

Thai speaking cabin attendants. This is clearly a leg qualification. If all 15 cabin attendants on the flight fly the same pairing we could apply the qualification constraint on the pairing instead of the leg. Since the number of tasks is generally much smaller than the number of legs this will cause fewer constraints in the master problem. Unfortunately the 15 cabin attendants may not fly the same pairing. Assume that 12 continue on the plane from Bangkok to Singapore (this leg only requires 12 cabin attendants) whereas 3 cabin attendants have a layover in Bangkok before they return to Copenhagen. Now the question is, how to "divide" the 5 language requirements between the 12-person pairing and the 3-person pairing? Since this question cannot be answered a priori, it must become a part of the optimization in the master problem and hence be modeled on the leg level.

Global constraints. All vertical constraints considered so far are applied on all (or some) tasks. For example, all pairings may have a language requirement. Each language constraint is applied on a small subset of all rosters, namely those, which contain the pairings in question. Global constraints are constraints, which put requirements on the entire solution. Examples of global constraints include:

- *Upper bound on the costs of the solution.* For example, one may wish to maximize crew satisfaction subject to a constraint on the total costs. Gamache et al. (1998) describe a version of this where the objective is to maximize crew bids subject to constraints on the number of "low time schedules" (which are unattractive) and the number of "open schedules," i.e., unassigned activities.
- *Constraints on overall bid satisfaction.* For example, 80% of all priority one bids must be granted or 80% of all days-off bids must be granted.
- *Horizontal rules defined on more than the planning period.* Examples: (i) one extra day off must be assigned every quarter of a year or (ii) at least one pairing to Japan must be assigned every year. Here a global constraint can help ensure that a sufficient number of extra days-off are assigned or that a sufficient number of crew members have been in Japan.

Further vertical constraints. The following constraints can be modeled in different ways and seen as composition of the building blocks described above.

- *Fly below rank.* In many places, a captain is qualified to fly as first officer, i.e., to fly below rank, but a first officer is rarely qualified to fly as a captain. Therefore airlines can increase the flexibility in the planning by having more captains and fewer first officers. There may be limitations on the assignment of first officer duties to captains. For example, particular captains may never be assigned such duties, such as newly promoted captains (who need to practice their captain qualifications) and very senior captains. Also there may be limitations on the number of first officer duties that can be assigned to other captains. Fly below rank occurs mostly on the pairing or task level.

- *Training programs*. In training programs, which are also referred to as route instructions, an instructor and a trainee have to fly in a certain part of the planning period together. A typical example is when a new pilot is getting his/her inflight training under the supervision of an instructor and an optional safety pilot. The trainees are supposed to fly together with several instructors during their training program, and with each instructor a certain number of consecutive flights.

2.2.3. Artificial rules

Usually, airlines introduce additional constraints which are not required due to legislation or contractual agreements, but which affect the quality of the schedule. Artificial rules (also called quality rules) are based mainly on the experience of the planners and provide a further restriction of the solution space such that unattractive solutions are omitted. Thereby, the following aspects are considered:

- Robustness of the solution.
- Support for solution methods.

At day of operation, the schedule might have to be changed due to changes in the timetable or to sickness of crewmembers. Hence one tries to take this into account at the planning stage by creating robust solutions. For example, if the maximum number of block hours in a seven day interval is 30, and in a particular roster one reaches 29 hours, it is quite likely that the roster becomes infeasible at a certain stage, since there is only a buffer of 60 minutes delay. Another example for planning with some margin is to permit layovers of 12 hours at day of operation but require 13 hours in the crew rostering problem. One more possibility is planning sufficiently many reserve duties.

Additional rules can also contribute to obtain better performance of solution methods by excluding rosters, e.g., from the generation, which are unlikely to be part in a good solution. For example, even if a rest of 10 days between two pairings might be legal, it would not be acceptable in practice and therefore it should be prohibited that these rests are considered and generated.

2.3. Objectives

There are four kinds of objectives which can be typically found in crew rostering problems. In most cases, airlines would apply some combination of them, either by combining them to one objective function, or by introducing global constraints for some of them while optimizing for another objective function.

2.3.1. Objectives related to real costs

Open time is the usual airline term for unassigned activities. The existence of open time does not necessarily mean that the solution is not feasible. Even if the unassigned activity is a pairing, it may be possible to solve the problem by asking crew to do over-time, by reducing the number of reserves, by hiring new crew or by canceling the flight. To minimize (often avoid) open time is always a major component of the objective function in crew rostering. Typically the cost of not assigning a task can be calculated a priori. It is often related to the type of task (ground duty or flight duty and crew category) as well as the length of the task. It is usually better to have three unassigned one-day pairings than one unassigned three-day pairing.

Overtime payment is another common example of a real cost component. In Europe, crewmembers are often paid a regular salary for up to a certain number of block hours per year. Block hours in excess of this have to be paid separately. In this situation one wants to avoid that some crewmembers receive overtime payments while others fly substantially less than what they can fly without extra payment. Since this cost is calculated on a yearly base, an approach similar to the one discussed under "equal assignment" (see below) must be applied.

A related problem concerns so-called "freelancers." These are pilots, that do not have a fixed salary and get paid for how much they fly. In this case, one tries to minimize the work assigned to freelancers while distributing the work among other crewmembers.

Other important cost components are usually related to simulator assignments. If simulator training is away from base, positioning flights (also called deadhead flights), hotel costs, etc., need to be considered.

Multibase problems may also contain a real cost component. Moving pairings between bases requires deadheads for crew. Crewmembers that are temporarily moved to another base may receive extra compensations.

2.3.2. Objectives related to the robustness of the solution

An alternative to "planning with margins" (as described above) is to permit, but penalize, patterns which may cause problems. Clearly, the cost must be related to the probability of "problems" multiplied by the excess cost of the "problem." Rosters which are likely to cause problems are generally those which "go to the limit" of some rule, but other issues may be relevant as well. For example, the reserve duty allocation should be compatible with the pairings. If most reserve blocks are one day long, but most pairings are intercontinental, the reserve blocks are not going to be useful in practice.

It is also well known within airlines that crewmembers assigned "hard" rosters are more likely to report sick than crewmembers assigned "easy" rosters. Often, this problem applies to particular crewmembers. This kind of knowledge is often problematic to formalize, but should be taken into account in the formulation of the objective function by penalizing "hard" rosters.

2.3.3. Objectives related to particular roster attributes

In most cases, various quality criteria are also modeled in the objective function. The criteria correspond to particular attributes of each assignment and how these are distributed among the crewmembers.

In Europe, equal assignment is predominant, which means that these attributes should be assigned as equally as possible. Examples include the number of duty hours, block hours, days off, per diem, early starts and layovers at particular airports. Also bid satisfaction may be an issue to be equalized. Often equalization within one planning period is not an important issue, but equalization over the year is the main issue. To accomplish this one has to keep track of historical data as well as data on planned absences, vacation, etc. Based on these data one can calculate personal targets for each crewmember for the current planning period for each attribute to be equalized. To implement the equal assignment criteria it is necessary to define the penalties for deviations from the average values. This function is nonlinear in general since a few large deviations may be worse than many small deviations.

Particular work patterns are also considered in rostering, in particular in the bidlines approach. Regularity in bidlines is considered very attractive. For instance, a "perfect roster" contains essentially the same pairings that are flown on the same weekdays throughout the roster. Since rostering is based on seniority in North America, unattractive patterns, such as night flying and one-day working periods, are condensed in as few rosters as possible that are then flown by junior crew.

2.3.4. Objectives related to individual preferences

In an increasing number of airlines, crewmembers can express preferences with respect to their schedule before the crew rostering step in the planning process. Different types of individual preferences ranging from bids for specific pairings to general scheduling preferences such as wishes for morning duties can be incorporated in the crew rostering problem.

Depending on the preferential bidding policy, awarding of the bids is done in strict seniority order or based on fair-share. Some airlines apply a combination of both, i.e., certain bid types are distributed equally while for others seniority is considered.

As mentioned in the introduction, a preferential bidding system draws large attention from crew and becomes very often a large political issue. Expectation management is of great importance in all stages of a production taking. For example, the Carmen Preferential Bidding system tackles this issue by providing a web based bidding interface for crew that presents crew with immediate feedback while they bid. In particular, the system determines and presents roster characteristics based on the preferences entered by a particular crewmember.

However, from a mathematical point of view preferential bidding can be mostly reduced to a different objective function while most of the general crew rostering principles apply.

3. Mathematical model

In this section we model the crew rostering problem mathematically. We first introduce a basic and simplified form of the problem and then extend it subsequently.

As pointed out above, most of the methods for crew rostering used in practice are based on the generate-and-optimize principle. Therefore we assume in the modeling that horizontal rules are implicitly satisfied by the roster generation. For further details we refer to section 4.1. In this section, we focus therefore on the modeling of vertical constraints as they occur in the master (optimization) problem and treat the generation problem as a black box returning legal rosters.

3.1. Basic model

Let us consider the crew rostering problem in its simplest form. Given is a set of activities $\mathcal{A}$ containing assignable activities such as pairings, ground duties and reserves, and a set of crew members $\mathcal{C}$. We denote the cardinalities of the two set by $m_{\mathcal{A}}$ and $m_{\mathcal{C}}$, respectively. The problem in this basic form is to obtain legal rosters $\mathcal{R}_k \subset \mathcal{A}$ for each crewmember k $(1 \leqslant k \leqslant m_{\mathcal{C}})$, which partition $\mathcal{A}$, i.e.,

$$\mathcal{A} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdots \cup \mathcal{R}_{|\mathcal{C}|},$$

and minimize some linear objective function on the costs of each roster c_k. As stated already, a roster is considered legal if it satisfies the horizontal rules and regulations.

In this form, the crew rostering problem can be viewed as a set partitioning problem with extra constraints on the subsets. A further particularity in the view of scheduling problems in general is that the scheduled activities are fixed in time.

When applying a generate-and-optimize approach, a large number of feasible rosters $\mathcal{R}_j \in \mathcal{A}$ (at least one per crewmember) are generated. Ideally, we would generate all feasible rosters, but for most major airlines this number would be enormous. After the generation of, say n, rosters, a set partitioning problem is solved for selecting the best combination of rosters with respect to the (linear) objective function.

Let $x \in \{0, 1\}^n$ be the decision variable whose jth entry is one if roster $\mathcal{R}_j$ is chosen, and let the costs of the rosters be given by $c \in \Re^n$. We denote by the $\{0, 1\}$ matrix A the $m \times n$ constraint matrix of the set partitioning problem, where $m = m_{\mathcal{A}} + m_{\mathcal{C}}$. The jth column of A has a one in row i if it contains the roster for crewmember i, and a one in row $k \in \{m_{\mathcal{C}} + 1, \ldots, m\}$ if activity $k - m_{\mathcal{C}}$ is part of the roster. In compact form, the set partitioning problem can be written as

$$\text{(SPP)} \quad z^* := \min\bigl\{c^t x\colon\ Ax = e,\ x \in \{0, 1\}^n\bigr\}, \tag{1}$$

where e denotes the vector of all ones. For each crewmember exactly one roster has to be assigned and each activity must be in exactly one roster.

We now introduce an example for the constraints in the basic model which is going to be extended in the following parts of the problem description.

Example 3.1. Suppose that for each crewmember three rosters have been generated, hence $n = 3m_{\mathcal{C}}$ and the constraint matrix A looks like as in table 1.

As mentioned above there are two types of constraints represented by A in the basic model. The *assignment constraints* make sure that each crewmember gets assigned exactly one roster, i.e., they form an $m_{\mathcal{C}} \times n$ block of A. The *activity constraints* ensure that each activity is assigned exactly once in the solution. The right-hand side is one in the basic model, i.e., each activity needs to be assigned exactly once.

Table 1

$Crew_1$			$Crew_2$				$Crew_{\mid\mathcal{C}\mid}$					
R_1	R_2	R_3	R_4	R_5	R_6	$\dots$	R_{n-2}	R_{n-1}	R_n			
1	1	1				$\dots$				=	1	Assignment
			1	1	1	$\dots$				=	1	constraints
						$\ddots$				$\vdots$	$\vdots$	
						$\dots$	1	1	1	=	1	
1	1	0	1	0	1	$\dots$	0	1	0	=	1	Activities
0	1	1	0	0	1	$\dots$	1	0	1	=	1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
1	0	1	0	0	0	$\dots$	0	1	0	=	1	

3.2. Extensions of the basic model

We now extend the basic model from the previous section and show how the various vertical rules introduced in section 2.2.2 are expressed. This is done by introducing additional constraints and by generalizing the set partitioning constraints.

Crew complement. Implementing crew complement in the basic model is straight forward. For each crew position under consideration, one $m_{\mathcal{A}} \times n$ block for the rosters is used. The right-hand side corresponds to the need on the particular position of the activity. Hence, we state the generalized set partitioning problem as

$$\text{(GSPP)} \quad z^* := \min\left\{c^t x \colon\ Ax = b,\ x \in \{0, 1\}^n\right\}, \tag{2}$$

where b denotes the vector of integer right-hand sides. The right-hand sides of the assignment constraints will all be ones whereas the right-hand sides of the activity constraints can take any positive integer value.

Example 3.2. To continue with the example introduced in section 3.1 let us consider that all activities require a crew complement of one captain (CP) and of one first officer (FO) except the last activity which requires two captains (this could be a simulator slot). Then the right-hand sides are all ones except for a two in the last row of the first block, while there are only ones in the second block representing the FO position (see table 2). The left-hand side contains again the rosters of the individual crewmembers. However, in each column of A only rosters for the crew position are given. In the example, the first two crew members are captains and the last one is a first officer.

Qualification constraints. As mentioned above, task- and leg-qualification constraints contain crew complement as a special case. For each task/leg where a certain qualification is necessary, an extra row per qualification is introduced. The right-hand side corresponds to the number of persons needed to have the particular qualification.

- *Inexperienced crewmembers.*

Table 2

Crew$_1$			*Crew*$_2$				*Crew*$_{\vert\mathcal{C}\vert}$					
CP			CP				FO					
R_1	R_2	R_3	R_4	R_5	R_6	…	R_{n-2}	R_{n-1}	R_n			
1	1	0	1	0	1	…	0	0	0	=	1	CP activities
0	1	1	0	0	1	…	0	0	0	=	1	
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	
1	0	1	0	0	0	…	0	0	0	=	**2**	
0	0	0	0	0	0	…	0	1	0	=	1	FO activities
0	0	0	0	0	0	…	1	0	1	=	1	
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	
0	0	0	0	0	0	…	0	1	0	=	1	

Table 3

Crew$_1$			*Crew*$_2$				*Crew*$_{\vert\mathcal{C}\vert}$					
experienced			inexperienced				inexperienced					
R_1	R_2	R_3	R_4	R_5	R_6	…	R_{n-2}	R_{n-1}	R_n			
0	0	0	**1**	0	**1**	…	0	**1**	0	$\leqslant$	**1**	Inexperienced
0	0	0	0	0	1	…	1	0	1	$\leqslant$	1	
⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	
0	0	0	0	0	0	…	0	1	0	$\leqslant$	1	

Example 3.3 (Inexperienced crewmembers). For the cockpit example from above, one now requires that for all activities at least one of the two pilots is experienced, i.e., at most one is inexperienced. This situation can be modeled in the following way (see table 3). For each experienced pilot, all the entries corresponding to this block are zero. For the inexperienced crewmembers, however, all activities in the roster on which the qualification constraint is applied are represented by ones. Since there can be at most one inexperienced pilot assigned to one activity, $\mathcal{R}_4$ and $\mathcal{R}_{n-1}$ cannot be in the solution at the same time since both crewmembers are inexperienced.

- *Must fly together.* This constraint specifies that two crew members, say *Crew*$_1$ and *Crew*$_{|\mathcal{C}|}$, must be assigned to the same activities. This constraint typically applies to a subset of the activities, e.g., the activities within a certain time interval. We need a constraint for each activity on which the constraint is applied. The constraint ensures that either both *Crew*$_1$ and *Crew*$_{|\mathcal{C}|}$ are assigned to the activity or none of them are assigned to the activity. This is done with a $+1$ and -1 coefficient, respectively, in case the activity is on the roster. Note that the number of must fly together constraints becomes quite large if there are many crew members who must fly together as we need one constraint per activity. Further, the empirical observation is, that the integer properties of these constraints tend to be poor, i.e., the integer program becomes much harder.

Table 4

$Crew_1$ fly together			$Crew_2$ fly together				$Crew_{\vert\mathcal{C}\vert}$					
R_1	R_2	R_3	R_4	R_5	R_6	$\dots$	R_{n-2}	R_{n-1}	R_n			
$+1$	$+1$	0	0	0	0	$\dots$	0	-1	0	$=$	0	Must fly together

Table 5

$Crew_1$ incompatible			$Crew_2$				$Crew_{\vert\mathcal{C}\vert}$ incompatible					
R_1	R_2	R_3	R_4	R_5	R_6	$\dots$	R_{n-2}	R_{n-1}	R_n			
1	1	0	0	0	0	$\dots$	0	1	0	$\leqslant$	1	Incompatibility

Table 6

$Crew_1$			$Crew_2$				$Crew_{\vert\mathcal{C}\vert}$					
R_1	R_2	R_3	R_4	R_5	R_6	$\dots$	R_{n-2}	R_{n-1}	R_n			
$\bar{c}_1$	$\bar{c}_2$	$\bar{c}_3$	$\bar{c}_4$	$\bar{c}_5$	$\bar{c}_6$	$\dots$	$\bar{c}_{n-2}$	$\bar{c}_{n-1}$	$\bar{c}$	$\leqslant$	C	Global constraint

Example 3.4. In our example, $Crew_1$ and $Crew_{|\mathcal{C}|}$ must either both be assigned to the first activity or none of them is assigned to the first activity (see table 4). The entries for the first and the last crewmembers of this row are $+1$ and -1, respectively, and the right-hand side is zero.

- *Incompatibilities*. Modeling incompatibilities is straight forward, too. For each of these restrictions one row is added, having all ones for the crew members for which the incompatibility applies. In general, this is a roster-qualification constraint.

Example 3.5. Regarding the example, suppose that the first and the last crewmembers must not fly together on the first pairing (see table 5). The entries for the first and the last crewmembers of this row are one, and the row sum must not be langer than one.

Global constraints. A typical global constraint is a bound on the overall costs as mentioned above. Suppose that for each roster we have costs $\bar{c}_j$ associated and that the modeled solution costs should not exceed some constant C.

Example 3.6. For our cockpit crew problem, this can be modeled by the extra row (see table 6).

Table 7

$Crew_1$ CP				$Crew_2$ CP				$Crew_{\|C\|}$ FO					
R_1^1	R_1^2	R_2	R_3	R_4	R_5	R_6	…	R_{n-2}	R_{n-1}	R_n			
1	0	1	0	1	0	1	…	0	0	0	=	1	CP rosters
0	0	1	1	0	0	1	…	0	0	0	=	1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	
1	1	0	1	0	0	0	…	0	0	0	=	1	
0	**1**	0	0	0	0	0	…	0	1	0	=	1	FO rosters
0	0	0	0	0	0	0	…	1	0	1	=	1	
⋮	⋮	⋮	⋮	⋮	⋮	⋮		⋮	⋮	⋮	⋮	⋮	
0	0	0	0	0	0	0	…	0	1	0	=	1	

Further vertical rules

In the following, we construct more complex constraints using the building blocks described above.

- *Fly below rank.* If one thinks of each (pairing, crew position)-combination as a unique task, then fly below rank belongs to the individual rule set. Alternatively the horizontal rule set may assume that a complement is always assigned in his usual position. In this case one will have to treat fly below rank as a relaxation of the vertical constraint requiring each crew position on the pairing to be assigned a particular number of times.

In the following example, two alternatives for the modeling of fly below rank are presented and their advantages and disadvantages are discussed.

Example 3.7. Let us consider the cockpit example again. Now assume that the first crewmember who has captain rank can fly below rank. As pointed out above, there are two options for modeling this. The first one is to use crew complement for this. The idea is to produce extra columns for this pilot which are basically copies of the existing ones except that one or more activities are generated for the FO position. If, in the present example, we restrict the pairings for which this can be the case to the first one then we generate extra columns for the particular crewmember, i.e., one extra column for the first crewmember (as shown for one roster in table 7). The disadvantage in this approach is the large number of extra variables constructed if there are many pilots that can fly below rank. However, this can be partly avoided by limiting the number of copies of rosters for a particular crewmember.

An alternative is to model fly below rank with qualification constraints and at the same time to use a different model for crew complement. For cockpit crew, for example, the modified crew complement constraint requires that there are two pilots assigned to the pairing. The qualification constraints take care of the fact that there is at least one crewmember with captain qualification. The advantage here is that the number of variables, i.e., columns is not increased (see table 8). However, after solving the IP, it is

Table 8

$Crew_1$			$Crew_2$				$Crew_{\lvert\mathcal{C}\rvert}$					
CP			CP				FO					
R_1	R_2	R_3	R_4	R_5	R_6	…	R_{n-2}	R_{n-1}	R_n			
1	1	0	1	0	1	…	0	1	0	=	2	Modified crew complement
0	1	1	0	0	1	…	1	0	1	=	2	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
1	0	1	0	0	0	…	0	1	0	=	2	
1	1	0	1	0	1	…	0	0	0	$\geqslant$	1	Qualification constraints
0	1	1	0	0	1	…	0	0	0	$\geqslant$	1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
1	0	1	0	0	0	…	0	0	0	$\geqslant$	1	

not determined who works in which rank. In practice, there are often limitations in how often a person can be assigned below rank over a certain time period. These rules cannot be modeled properly with this approach.

- *Training programs.* As introduced in the previous section, route instruction is the requirement that two or three crewmembers must be assigned to the same trips during a certain period. In the Carmen Crew Rostering systems, this is modeled using one must-fly-together constraint for each task in this period. One of the three crewmembers, e.g., the instructor, has a $+2$ coefficient and the others have -1 coefficients in the constraint. The generation method provides the right coefficients to consider only activities falling into the period and on top of that generates “similar” rosters for these groups.

4. Solution methods

4.1. A commercial crew rostering system

In this section we first give a brief presentation of the Carmen Crew Rostering system as an example of how to solve the crew rostering problem in practice. In the subsequent sections we detail the algorithms and methods employed in the system.

Figure 2 shows the three main components of the automatic planning functionality of the Carmen Crew Rostering system: the rule evaluator, the generator and the optimizer. The purpose of the generator is to generate a large number of possible rosters for each crewmember. The optimizer chooses one roster per crewmember while ensuring that all pairings and other duties are assigned and vertical constraints are respected. The generator uses the rule and value evaluator to ensure legality of the rosters generated and to calculate the other attributes of the generated rosters.

The core of the automatic solution methods is the generator. The key to getting the best possible solution is – not surprisingly – to generate the right rosters. Most of our

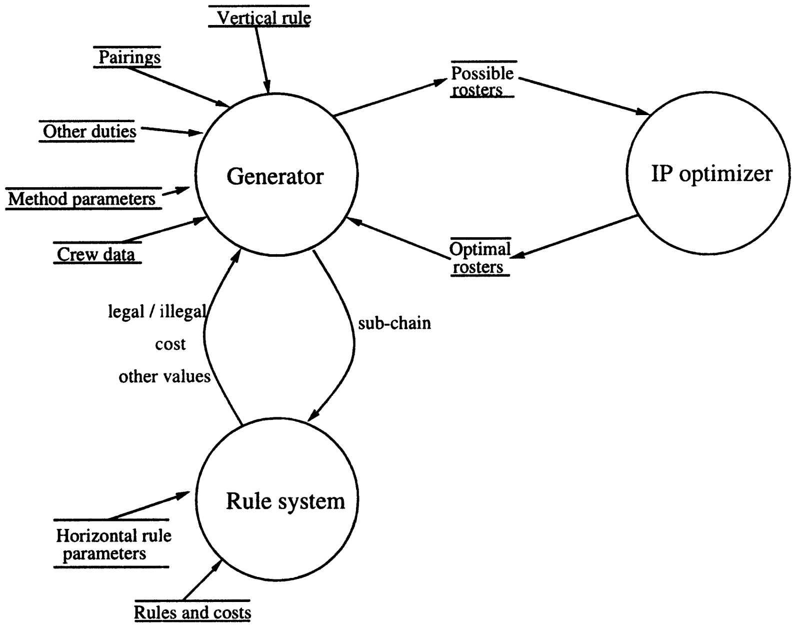

Figure 2. Overview of the Carmen Crew Rostering system.

efforts on automatic planning are put into the development of generation methods. The generator is covered in depth in the following sections. Here we briefly discuss the rule evaluator and the optimizer.

The rule modeling tool *Carmen Rave* is a special purpose programming language used to express the horizontal rules and regulations. It is also used to specify the objective function coefficients and other attributes of a roster or a chain of activities. Such values can be the contribution of the roster to a vertical constraint or a value used to control the generation process. The Rave source code is compiled into C code by the Carmen Rave Compiler supplied with the Carmen system. This makes it possible for the end user to maintain and modify rules and regulations.

The concept of a rule programming language is not unique for the Carmen systems. The crew scheduling products developed by SBS (SBS International, 2003) also come with a rule programming language *Rule Talk* which is somewhat similar to Carmen Rave. To our knowledge, other vendors do not provide a rule programming language; however, table and parameter driven configuration of rules is supported by their products.

There are a number of advantages to a rule modeling language. These include:

- Fast and correct modeling of the problem. The user can read and understand the rule implementation. This makes it possible to verify correctness without exhaustive testing.
- It is easy to change and maintain the rules, regulations and objectives. The user can do it himself/herself.

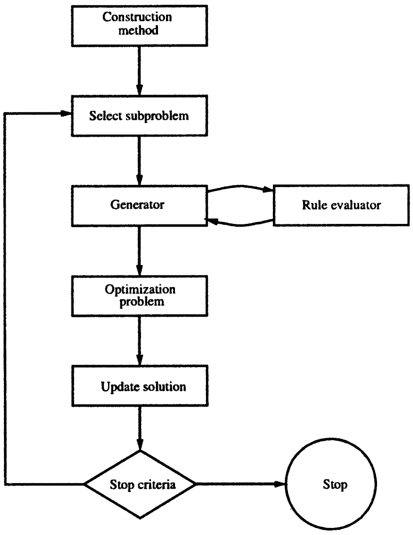

Figure 3. A typical solution process.

- Consistency between rules used in crew pairing, crew rostering and preferential bidding. The same rule implementation can be used.

However, these features do not come for free. The optimization methods, that is the generator and IP optimizer components in figure 2, have no particular knowledge of the rules and objectives and can only “ask” simple questions about legality and cost. Therefore some problem-specific knowledge cannot be used in the solution process.

Problems of type (2), with the extentions described in section 3 are integer programs for which any standard IP solver can be used. In the Carmen Crew Rostering system we had originally used ILOG CPLEX (ILOG, 2002) and have recently shifted to Dash Optimization’s Xpress-MP (Dash Optimization, 2002). We have done a limited tuning of parameters including focus on feasibility rather than proven optimality and choice of LP optimizer (simplex or interior point) depending on the problem size.

4.2. Construction and improvement methods

We distinguish between construction methods, which are used to construct an initial solution from “scratch” and improvement methods which can be applied iteratively to improve a given solution. Figure 3 illustrates the usual solution process. First a construction method is used, then improvement methods are applied iteratively. The methods can be configured in numerous ways through parameters and expressions in the rule language. They can be run in any order and terminated at any point, keeping the best solution obtained.

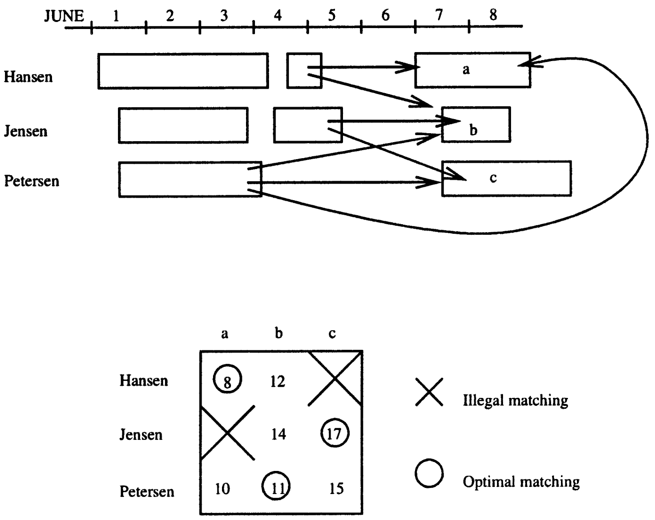

Figure 4. The Carmen construction method.

4.2.1. Construction methods

The main construction method is based on sequential assignment of activities to crewmembers. In each iteration of the construction method, an optimal assignment (1-matching in a bipartite graph) of (at most) one activity per crewmember is constructed. If an activity must be assigned to more than one crewmember it is simply duplicated in the graph. However, we do not need to evaluate legality and costs for assigning each of the different copies of the same activity to the same crewmember.

Solving the assignment problem and updating the rosters may leave a number of activities unassigned, so the procedure is applied iteratively. In each iteration the starting point is the rosters constructed so far and the so far unassigned activities. The method terminates when all activities have been assigned or when no more assignments are legally possible. Figure 4 illustrates one iteration of this process. The upper part shows the possible assignments. The lower part shows the costs associated with each legal assignment. Note that in this example the optimal assignment of activities to crewmembers requires that only one crewmember get his/her "best" activity.

Note that the objective function in the construction method needs not be the "real" objective function. For example, very long activities may be very important to assign initially since they will be difficult to assign with the improvement methods applied later in the solution procedure. Such consideration can be incorporated in the objective function of the construction heuristic.

4.2.2. The path graph and the individual possible set

Before the execution of any improvement method a *path graph* is set up. The path graph $\mathcal{G}$ is a directed acyclic graph with node set $\mathcal{A}$ (the set of activities) plus a source and a sink node and arc set corresponding to the possible combinations of activities which can follow immediately after each other. The existence of arc (a_1, a_1), $a_1, a_2 \in \mathcal{A}$, does not guarantee that activity a_2 can always be assigned as immediate successor of activity a_1, but the absence of the arc guarantees that it will never be possible.

The existence of an arc is determined by evaluating legality on a roster consisting of the two activities assigned to an "anonymous" crewmember. To set up the entire graph requires slightly less than $|\mathcal{A}|^2/2$ rule evaluations since we only have to consider the existence of arc (a_1, a_2) if a_2 starts later than the end of a_1. The reason for using an "anonymous" crewmember is to save memory. With, for example, 2000 crewmembers and $|\mathcal{A}| = 6000$ an explicit representation of individual path graphs could contain as much as 36 billion arcs whereas the "anonymous" path graph contains at most 18 million arcs.

The path graph is not necessary to generate rosters, but should be seen as a cache of information about the compatibility of pairs of activities. Without this cache many unnecessary rule evaluations would have to be carried out.

Since the path graph is not personal it is not suitable for storing information on the (crewmember, activity) compatibility. Often an activity cannot be assigned to some crewmembers since special qualifications are required. It is important to cache such information instead of "discovering" it at the expense of a rule evaluation. To accomplish this, one can set up an explicit representation of the set of possible activities for all crew members. This is also done once before generation starts and requires $|\mathcal{C}||\mathcal{A}|$ rule evaluations. The individual possible set is used together with the path graph. During generation, if we consider to assign activity a_2 immediately after activity a_1 to crewmember c we test two conditions before executing a rule evaluation:

1. Is a_2 in the possible set of c?
2. Does the arc (a_1, a_2) exist in the path graph?

4.2.3. Generation limitations

As opposed to the set up of the path graph and the individual possible sets, the generation limitations are carried out in each iteration of the improvement method. The starting point is the solution rosters from the previous iteration, i.e., those picked by the IP optimizer. For each crewmember we choose a subset of those activities found on the solution roster. These activities are considered "locked" in the current iteration – they belong to the required set. The remaining activities are deassigned from the roster and become possible for all crewmembers. The number of rosters which can be generated depends heavily on:

1. The number of required activities on a roster. More required activities means less time for possible activities which means fewer generated rosters.
2. The number of possible activities. Many possible activities means many generated rosters.
3. The duration of the possible activities. Short activities means that more activities will fit on a roster which, in turn, increases the number of generated rosters.

The user can influence the size of the first two measures and the selection of the required set through method parameters as we are going to discuss in section 4.2.5.

4.2.4. The generator

When generation starts, a crewmember has a (possibly empty) set of required activities, i.e., activities which must be assigned. These activities may be "locked" to the crewmember by the user or may be locked temporarily, cf. section 4.2.3. The set of possible activities consists of all those activities which are currently not locked to a crewmember. It includes also those activities which have been unlocked for this iteration.

For a particular crewmember the set of possible activities can be further reduced to those that "can be reached in the path graph" from the required activities. For example, activities overlapping time-wise with those in the required set need not be considered. We need not treat this explicitly, since the generator only considers incremental augmentations corresponding to arcs in the path graph.

The generator considers all crewmembers sequentially. For each crew member it carries out a depth first search. For the sake of simplicity we will consider the case where there is only required activities R before all possible activities P, i.e., we are generating the last part of the roster. In this case the generator will first find the last required activity *last* and execute the recursive function generate:

```
generate($R$, $last$)
if legal($R$) then
begin
  if complete($R$) then output($R$)
  $\forall next \mid (last, next) \in \mathcal{G} \wedge next \in P$
  begin
    $R := R \cup next$
    generate($R$, $next$)
    $R := R \setminus next$
  end
end
```

The generator uses incremental rule evaluation to prune the search as soon as the roster becomes illegal (the recursive call of generate initially tests that legal is true). This is to avoid generating a large number of rosters which are all illegal due to choices done early in the search. Incremental rule evaluation is crucial for performance, but introduces the so called *illegal subchain* problem. This occurs if, for example, activities a, b and c constitute a legal roster, but activities a and b (without c) make the roster illegal. In this case the incremental rule evaluation will prune the search after a and b have been assigned, so the roster consisting of a, b and c will never be generated. Typical examples of such rules are rules which require, for example, working time or working days to be at least something.

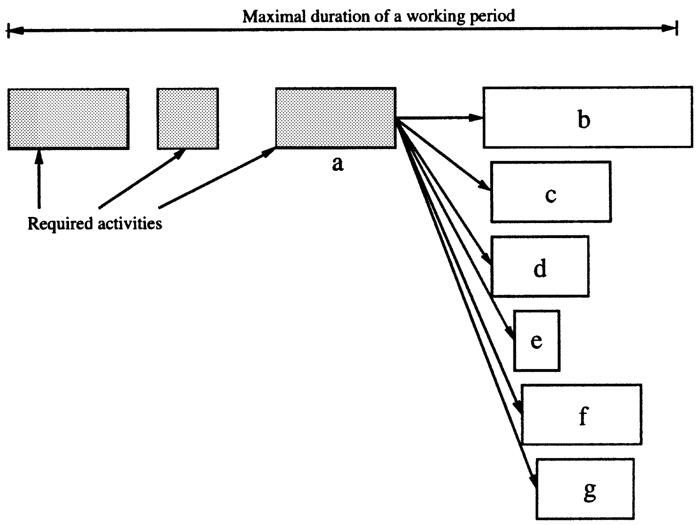

Figure 5. The possible continuations of a path. The shaded activities are required. The white ones are possible.

To overcome the illegal subchain problem the rule programmer can state that a rule should only be used to evaluate when a roster is *complete.* This is what we need the function complete for. A roster may be legal without being complete. This is the case if the roster violates a rule with the illegal subchain property, but satisfies all other rules. In this case we do not want to output (the function output) it to the IP optimizer, since it is not a legal roster from the users point of view, but we do not want to backtrack in the generation either, since it may be the starting point of legal rosters.

The user can limit the search width through method parameters. With a search width of, e.g., four, only four legal arcs with their origin in a node $a \in \mathcal{A}$ are considered. Here legal means that the node (activity) at the end of the arc can legally be inserted in the roster of the current stage of the search, i.e., legal returns true. This can be illustrated with an example.

In figure 5 the arcs (a, b), (a, c), (a, d), (a, e), (a, f) and (a, g) all exist in the path graph, but with the current set of required activities and the limit on the duration of a working period, b cannot legally be assigned. Instead we consider the first 4 legal arcs, corresponding to the assignment of activities c, d, e and f, respectively. Should it turn out that the roster ending with $-a - d$ is illegal due to some other rule, the generator would have considered the arcs (a, c), (a, e), (a, f) and (a, g).

This discussion suggests that the set of arcs with their origin in activity a (or any other activity) must be ordered. In figure 5 they are ordered by starting time. This is the default ordering, which makes sense in many cases because assigning the first possible activity will minimize the unproductive time between activities. However, in some cases a different ordering would make sense, and for that reason the rule programmer can implement one or more sort orders. We give some examples where it is appropriate to define another ordering.

- Assume that the maximal duration of a working period is five days and the minimal rest between two working periods is 60 hours. In this case it is efficient to assign an activity yielding a short rest or assigning an activity yielding a rest of slightly more than 60 hours. If a minimal rest in all cases must be 10 hours an appropriate variable to sort by is rest time minus 60 hours if rest time is at least 60 hours and rest time minus 10 hours, otherwise.
- Assume that pairings with flight legs to and from Rio de Janeiro require some Portuguese speaking crewmembers (vertical constraint) and that Portuguese crew speaking crewmembers is a scarce resource. Therefore we should prefer the Rio pairings over other pairings when generating for Portuguese speaking crewmembers.

4.2.5. The crew improve method

We will now give examples of some of the concepts discussed in this section by considering a slightly simplified version of the so-called crew improve method, which is one of the main generation methods in the Carmen Crew Rostering system. The discussion is not intended to cover all parameters and control possibilities of this method and is just one of several methods found. The intention is to give a flavor of how the system can work.

The crew improve algorithm.

```
$\forall c \in \mathcal{C}$ calculate $Group(c)$
for $It1 := 1$ to $It1_{\max}$
begin
  $\forall groups\ g$
  begin
    $tw_{\mathrm{cur}} := tw_{\mathrm{init}}$
    for $It2 := 1$ to $It2_{\max}$
    begin
      $\forall c \mid Group(c) = g$
        Remove all activities $a \in \mathcal{A}$ for which $Overlap(a, tw_{\mathrm{cur}})$
        from the set of required activities of $a$
      $\forall c \in \mathcal{C}$: generate
      Solve the integer program and update the solution
      $tw_{\mathrm{cur}} := tw_{\mathrm{cur}} + tw_{\mathrm{step}}$
    end
  end
end
```

Group is a mapping from a crewmember $c \in \mathcal{C}$ to an integer defined by the rule programmer, which partitions the set of crewmembers into groups for which $Group(c)$ is identical. As a special case all crewmembers may be in the same group. Generally crewmembers for which $Group(c)$ is identical have "something in common," e.g., the same crew rank (position). This is motivated by the fact that crew members belonging to

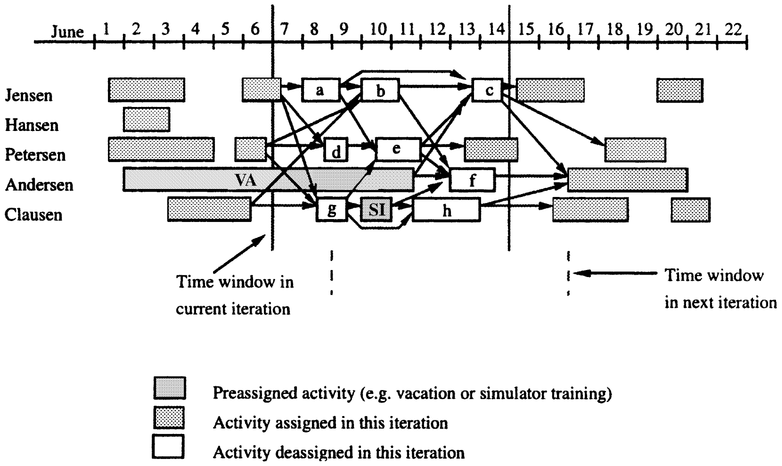
June
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Jensen
Hansen
Petersen
Andersen
Clausen
a
b
c
d
e
f
g
h
SI
VA
Time window in current iteration
Time window in next iteration
Preassigned activity (e.g. vacation or simulator training)
Activity assigned in this iteration
Activity deassigned in this iteration

Figure 6. The opening of activities in a time window.

the same rank "compete" for the same activities, so they should be optimized together, whereas the different crew ranks are only coupled through the vertical constraints they contribute to.

The current time window and the initial time window denoted tw_{cur} and tw_{init} respectively are time intervals given by a start and end time. The initial time window is given by the user in the parameter form, whereas the current time window changes during the course of the iterations. *Overlap* is a function which determines whether activity a falls within the time window tw_{cur}.

There are two iteration counters in the algorithm. $It1_{\max}$ determines how many times a crew group is optimized. Typical values are three to five. $It2_{\max}$ counts the number of improvement iterations within each group. This figure generally depends on the time window step size tw_{step} and the planning horizon.

Note that rosters are generated for all crewmembers. For those crewmembers where $Group(c) \neq g$, i.e. those which are not in the current group, the required set of activities is quite large, so it may not be possible to generate anything but the current roster. Therefore the user may, in the parameter form, choose to replace $\forall c \in C$ in the third "begin" block by $\forall c \mid Group(c) = CurGroup$.

The current time window is incremented by tw_{step}, that is its start and end times are incremented by this value. tw_{step} is a user-defined method parameter which can be changed in the parameter form.

Figure 6 illustrates the crew improve methods for a case where $Group(c)$ is identical for all c.

4.2.6. The k-shortest path generator

Much recent academic work on crew rostering and related problems, see, e.g. (Desaulniers et al., 1998) for a comprehensive review of this approach, has focused on the Dantzig–Wolfe decomposition or reduced cost column generation approach, where the generation of columns, in our case rosters, is a shortest path type optimization problem known as the shortest path problem with resource constraints. With this approach it is, at least in theory, possible to prove lower bounds and in certain cases even obtain proven optimal solutions. Impressive computational results have been obtained on a number of time constrained routing and scheduling problems, see (Gamache and Soumis, 1998; Gamache et al., 1998, 1999).

The shortest path problem with resource constraints is a generalization of the well-known shortest path problem. The one-dimensional cost label is replaced by a multi-dimensional label and several labels can be associated with each node of the graph. Each label corresponds to a distinct path and all relevant (with respect to costs and rule evaluation) attributes of this path are aggregated in the label. In a simple example the label could be the triple (block hours, working days, cost). Each element of the triple is referred to as a resource variable. To evaluate the legality and cost of a path, we only need to know the resource variables. The path itself is not of relevance. The resource variables are used for two distinct purposes:

1. The resource variables are used to test legality. Given the right resource variables it is sufficient to test that their value is within the legal domain. This is clearly most efficient if the number of resource variables is limited and if they can be calculated without explicitly considering the entire path.

2. The resource variables are used to compare two different (partial) paths, i.e., two different (partial) rosters. Given certain non-trivial properties of the rules and cost function one can establish the mathematical conditions under which one path dominates another path. This knowledge is exploited to prune the search along paths which are extensions of a path dominated by another path. This way it is sometimes possible to implicitly enumerate the entire space of possible rosters.

From a theoretical point of view the column generation approach is a beautiful application of two strong mathematical programming concepts: linear programming theory and dynamic programming. However, it is based on some assumptions which are rarely completely valid in practice. In particular, we point out the following drawbacks of the column generation approach:

- The transformation of a complex real-world rule set and objective function into resource variables is non-trivial and can only be done by experts. We are not aware of the existence of a rule language and compiler for this purpose.
- Most real-world problems do not posses the properties required by the column generation approach to work efficiently. Therefore the pruning motivated by dominance cannot be exploited, or can only be exploited heuristically. Thus, column generation reduces to an efficient way to test rules.

The column generation approach has been implemented in commercial crew scheduling systems used by a number of airlines, see, e.g. (AD OPT Technologies, 2003).

At the same time, there are ways to exploit column generation without sacrificing a rule programming language. In (Hjorring and Hansen, 1999) this is demonstrated and described in detail for the crew pairing problem. The method is denoted the k-shortest path method. In short, the idea is to solve a shortest path problem (no resource constraints). If the pairing corresponding to the shortest path happens to be legal it is clearly the shortest legal path. Otherwise (the usual case), the second shortest path is generated, legality is tested, etc. Sooner or later the shortest legal path will be generated this way. Finding the k node disjoint shortest path in an acyclic directed graph is computationally cheap, see, e.g. (Martins and Santos, 2000), while the expensive operation is to test legality.

In the Carmen Crew Rostering system the k-shortest path approach is also implemented. We solve the shortest path problem in the path graph with some activities in the required set, for example, chosen as in the crew improve method, cf. section 4.2.5. Then the rule evaluation code is executed on the path (roster). If it is not legal the second shortest path is found, etc.

The k-shortest path approach is in principle compatible with any rules and the Carmen Rave compiler. In practice it will only perform well, if there is a reasonable chance that a path will correspond to a legal roster. This chance is highly dependent on the set of required and the set of possible activities, so configuration of the search parameters is equally important with the k-shortest path approach as it is with crew improve approach. A draw back shared with the traditional reduced cost column generation approach is the restriction on the structure of the objective function. The objective function contribution of a roster must decompose by nodes and arcs, i.e., by activities and combinations of activities in sequence. This is often the case if the objective function measures the bid satisfaction of the rosters. It is also the case if the objective is to assign as many activities as possible to the crewmembers, i.e., to minimize open time.

4.3. Problem extensions

In this subsection we sketch the treatment of some more advanced real world problems which are not covered by the discussion above.

Treatment of infeasibilities

In practice one rarely knows whether all the constraints of the IP models of type (2) with extentions can actually be satisfied. There may be a shortage of crew members for the planned production, a shortage of certain qualifications or just errors in the data. In any case, the answer “your problem is not feasible” is not very useful or informative for the user. Therefore, we always add slack variables corresponding to the non-assignment of an activity or breaking a vertical constraint.

Base variants

The same activity may exist in several variants depending on the base of the crewmember carrying out the activity. For example, a pairing may have its first operating leg starting in Milan and its last active leg ending in Rome. It can be assigned to a crew member based in Milan by appending a deadhead (passive transport) from Rome to Milan. It can also be assigned to a crew member based in Rome by insertion of an initial deadhead. We refer to the two variants as base variants of the same pairing. Base variants might not be attractive at first sight since they will produce deadheads, but may be necessary if it is not possible to ensure the correct distribution of pairings between bases in the crew pairing problem.

Ground activities can also have base variants. If the flight simulator is located in Frankfurt, a flight simulator activity to be assigned to a Munich based crewmember must be a base variant with a deadhead at the beginning and end.

From an optimization point of view base variants are not a problem. The generator must know the base of the crewmember it is generating for to determine the right base variant of the activities in the possible set. Also one must account correctly for the cost, but there are no principal problems.

Strict seniority

Strict seniority is known in connection with preferential bidding where crewmembers can enter bids for certain activities and days-off. In some airlines the more senior crewmembers' bids are considered more important than less senior crewmember's bids. In extreme cases the most senior crewmember's bids are considered infinitely more important than the bids of the second most senior crewmember, and the bids of the second most senior crewmember are considered infinitely more important than the bids of the third most senior crewmember, etc. Strict seniority is mainly found in crew rostering problems in North America.

Mathematically strict seniority can be modeled by multiplying the objective function coefficients by very large numbers. If we are maximizing bid satisfaction, and bid satisfaction of a roster can be expressed by a number in the interval $[0, 100]$ then we can calculate the objective function coefficient for a roster generated for the most senior crewmember as the bid satisfaction multiplied by $101^{|\mathcal{C}|-1}$. For numerical reasons, this is not possible in practice for more than a handful of crew members.

A naive solution to the strict seniority problem is to optimize the roster of the most senior crewmember first, then lock all activities on this roster and optimize the roster of the second most senior crewmember, etc. This is time consuming since it requires the entire problem to be solved $|\mathcal{C}|$ times and it is highly sub-optimal since there will generally be many different rosters that give the maximal bid satisfaction. By selecting one (arbitrarily) of these we limit the possibility for granting the bids of less senior crewmembers.

A more clever solution was proposed in (Gamache et al., 1998). Essentially the idea is to impose constraints on the bid satisfaction of the more senior crewmembers, instead of selecting a particular roster. Depending on how this procedure is carried out, it is possible to obtain proven optimality. In the Carmen System this idea is implemented by solving the IP problem iteratively. First we optimize the bid satisfaction of the most senior crewmember. Then we impose an extra constraint requiring the bid satisfaction of this crewmember to be maximal. Then we resolve the IP maximizing the bid satisfaction of the second most senior crewmember, etc.

5. Practical experience

5.1. Problem solving approach

The challenge of real-world optimization is to find the best possible solution, to the right problem, as fast as possible. All these three dimensions are essential to the success of a crew rostering system. The academic research almost solely focuses on the first dimension in the case of "difficult" problems (e.g., large scale vehicle routing problems) and on the third dimension in the case of "easy" problems (e.g., linear programming).

The second dimension – the right problem – is too often omitted even in scientific work supposed to be related to airline problems. All commercial systems of today solve a problem which is not exactly the real life problem, but an approximation, or a subproblem resembling the real life problem only to a certain extent. Even though the subproblem may be solved optimally, the underlying real life problem is only partially solved. In fact, the crew rostering problem itself is not the "right problem." The real problem is to assign individual tasks to individuals. The pairings, which we have assumed to be input to the crew rostering problem are just the result of another suboptimal planning problem – the crew pairing problem. Even the crew rostering problem described in this paper is to some extent a simplification of the real problem faced after pairings have been produced. For example, complex training programs where pilots have to go through a series of phases, each with different requirements on the assigned instructor(s), duration and activities to be carried out are not really contained within the framework outlined above. By employing a rule modeling tool and generic and somewhat "low tech" generation algorithms, which do not make any particular assumption on the structure of the particular problem to be solved, the Carmen Crew Rostering system supports a sufficiently accurate modeling of the real-world crew rostering problem.

But the true real-world challenge is not to be able to model a 200 page specification reasonably accurate. The problem is that this specification does not exist. Rules and objectives must be explored and revealed iteratively and the result must be under continuous evaluation. A prototyping process with fast iterations is essential to get as close as possible to the real problem within an implementation project.

The development of rule modeling tools and languages has enabled airlines to significantly increase the level of detail in optimization models. Airlines now have the ability to include every hotel room, every positioning flight, every meal at actual cost in their optimization models – at least to the extent that these data exist. The tools are also used to model other quality criteria such as robustness, quality of service, and quality of life. All these refinements in the model have practically eliminated the need for manual interaction in the solution process for crew pairing and crew rostering.

It is important to realize that all three dimensions of performance – best possible solution, right problem, and as fast as possible – are linked in the sense that an improvement in one dimension will always be at the cost of some compromise in another dimension. The trade-off is largely an issue of configuration, and airlines tend to prioritize the costs and correctness aspects more than anything else, since this is where the biggest potential for improvement lies.

5.2. The use of the Carmen Crew Rostering system

The Carmen Crew Rostering system was taken into production first at KLM in 1995 and is as of now (May 2003) used in production at eight airlines in Europe and North America, and additionally at SJ, the Swedish State Railways and Deutsche Bahn, the German State Railways. The different requirements of these customers are met through the use of Rave, allowing detailed, airline specific installations of the systems.

The largest single production problems found at Carmen's customers have currently a population of about 1600 crew and more than 5000 assignable activities in a single planning unit. Another problem contains 7200 crew and 2000 assignable activities (of which most have to be assigned to several crew). Such problems solve usually in less than 10 hours, i.e., in an overnight run. However, the problem size itself is not the only factor influencing solution times. The complexity of the rules and objectives tends to be of greater importance and certain much smaller problems require substantially longer solution times.

It is generally very difficult to calculate the savings in crew rostering. Most of the real costs are considered during the pairing construction, leaving mainly the minimization of open time and overtime as real costs. However, non-tangible aspects such as quality of the rosters are as important, and especially in preferential bidding, the maximization of crew satisfaction. This is a main difference to the crew pairing problem. Crew pairing has a quite well-defined monetary objective function and different solutions are easily comparable. Consequently it is also easier to publish savings figures for crew pairing systems and to compare (benchmark) crew pairing systems against each other. In the case of a crew rostering system this comparison is typically more fuzzy.

During the various crew rostering production setting processes Carmen has carried out and is carrying out, we have learned that the crew pairing problem is cleaner and simpler and a purer optimization problem. In contrast to that, crew rostering problems are very airline specific. In the following, we point out some of the particularities of crew rostering compared to crew pairing:

- The complexity of rules is higher. Most pairing rules still apply in the crew rostering problem, but there are a number of rostering specific rules and it is necessary to handle exceptions applying to particular ground duties, to handle the transition from one planning period to the next as well as rules defined on a longer time interval than the planning period.
- The objectives are less clear and difficult to quantify. In particular it is difficult to scale them together. Also there tends to be a larger number of objectives which have not been formalized and must be explored and revealed iteratively during the customization of a crew rostering system. Often crew satisfaction is a significant part

of the objective, but it is not at all clear how to measure crew satisfaction. Granted bids must be translated into points which in turn must be translated into objective function coefficients. This must be done in a fair and logical way, and the process must also be comprehended and accepted by the crew!

- The implementation of a crew rostering system at an airline is a major political issue drawing large attention from the crew member unions. Expectations as well as concerns are high and managing these expectations and concerns is a project in itself. This is in particular the case for a preferential bidding system. In essence the point of a preferential bidding system is to make crew happy. Whether the crewmembers actually become happy depends more on the ratio between what they get and what they expected rather than on the magnitude of the numerator of this fraction.

6. Summary and outlook

We presented airline crew rostering as an important part of airline operations and an interesting problem for the application of operations research. We described not only how different types of crew rostering problems can be modeled but also how they are solved with commercial software systems, and presented the Carmen Crew Rostering system as example. We also highlighted practical considerations regarding the production setting of crew rostering systems.

In the future, crew pairing and crew rostering will be integrated into one planning problem. Only a combined problem represents the correct formulation of the crew planning problem, and provides the best rosters, both from a cost and a quality point of view. We view the creation of rosters out of legs, i.e., skipping the intermediate step of producing a pairing solution, as one of the important research areas in crew planning. Another area which currently attracts great interest is *crew recovery,* where after publication individual rosters are maintained and repaired until and on the day of operations.

Acknowledgments

We thank our colleague Dr. Curt Hjorring for his comments that improved an earlier version of this paper. Further we thank the many colleges at Carmen Systems who, during the years, have contributed to the development of the Carmen Crew Rostering system. We are also grateful to the anonymous referees for valuable feedback and comments.

References

AD OPT Technologies. (2003). `www.ad-opt.com`

Anbil, R., C. Barnhart, L. Hatay, E.L. Johnson, and V.S. Ramakrishnan. (1992). "Crew-Pairing Optimization at American Airlines Decision Technologies." In T.A. Ciriani and R.C. Leachman (eds.), *Optimization in Industrial Environments,* pp. 7–24. Wiley.

Anbil, R., E. Gelman, B. Patty, and R. Tanga. (1991). "Recent Advances in Crew-Pairing Optimization at American Airlines." *Interfaces* 21(1), 62–74.

Andersson, E., E. Housos, N. Kohl, and D. Wedelin. (1998). “Crew Pairing Optimization.” In G. Yu (ed.), *Operations Research in the Airline Industry,* pp. 228–258. Norwell, MA: Kluwer Academic.

Augustsson, L. (1997). “Partial Evaluation in Aircraft Crew Planning.” In *Partial Evaluation and Semantics-Based Program Manipulation*, Amsterdam, The Netherlands, June 1997, pp. 127–136. New York: ACM.

Barnhart, C., L. Hatay, and E.L. Johnson. (1995). “Deadhead Selection for the Long-Haul Crew Pairing Problem.” *Operations Research* 43(3), 491–495.

Barnhart, C., E.L. Johnson, R. Anbil, and L. Hatay. (1994) “A Column-Generation Technique for the Long-Haul Crew-Assignment Problem.” In T.A. Ciriani and R.C. Leachman (eds.), *Optimization in Industrial Environments*, Vol. 2, pp. 7–24. New York: Wiley.

Barnhart, C. and R.G. Shenoir. (1996). “An Alternate Model and Solution Approach for the Longhaul Crew Pairing Problem.” Technical Report, MIT.

Barutt, J. and T. Hull. (1990). “Airline Crew Scheduling – Supercomputers and Algorithms.” *SIAM News*.

Campbell, K.W., R.B. Durfee, and G.S. Hines. (1997). “FedEx Generates Bid Lines Using Simulated Annealing.” *Interfaces* 27(2), 1–16.

Cappanera, P. and G. Gallo. (2001). “On the Airline Crew Rostering Problem.” Technical Report TR-01-08, Department of Computer Science, University of Pisa, Italy.

Caprara, A., M. Fischetti, P. Toth, D. Vigo, and P. Guida. (1998a). “Algorithms for Railway Crew Management.” *Mathematical Programming* 79, 125–141.

Caprara, A., F. Focacci, E. Lamma, P. Mello, M. Milano, P. Toth, and D. Vigo. (1998b). “Integrating Constraint Logic Programming and Operations Research Techniques for the Crew Rostering Problem.” *Software, Practice and Experience* 28(1), 49–76.

Caprara, A., P. Toth, M. Fischetti, and D. Vigo. (1998c). “Modeling and Solving the Crew Rostering Problem.” *Operations Research* 46, 820–830.

Christou, I.T., A. Zakarian, J. Liu, and H. Carter. (1999). “A Two-Phase Genetic Algorithm for Large-Scale Bidline-Generation Problems at Delta Air Lines.” *Interfaces* 29(5), 51–65.

Dash Optimization. (2002). *Xpress-MP User’s Manual*. Northants, UK: Dash Optimization.

Dawid, H., J. König, and C. Strauss. (2001). “An Enhanced Rostering Model for Airline Crews.” *Computers and Operations Research* 28(7), 671–688.

Day, P.R. and D.M. Ryan. (1997). “Flight Attendant Rostering for Short-Haul Airline Operations.” *Operations Research* 45(5), 649–661.

Desaulniers, G., J. Desrosiers, Y. Dumas, S. Marc, B. Rioux, M.M. Solomon, and F. Soumis. (1997). “Crew Pairing at Air France.” *European Journal of Operational Research* 97, 245–259.

Desaulniers, G., J. Desrosiers, I. Loachim, M.M. Solomon, F. Soumis, and D. Villeneuve. (1998). “A Unified Framework for Deterministic Time Constrained Vehicle Routing and Crew Scheduling Problems.” In T. Crainic and G. Laporte (eds.), *Fleet Management and Logistics,* pp. 57–93. Norwell, MA: Kluwer Academic.

El Moudani, W., C.A. Nunes Cosenza, M. de Coligny, and F. Mora-Camino. (2001). “A Bi-Criterion Approach for the Airlines Crew Rostering Problem. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello, and D. Corne (eds.), *First International Conference on Evolutionary Multi-Criterion Optimization,* pp. 486–500. Lecture Notes in Computer Science, Vol. 1993. Berlin: Springer.

Fahle, T., U. Junker, S.E. Karisch, N. Kohl, M. Sellmann, and B. Vaaben. (2002). “Constraint Programming Based Column Generation for Crew Assignment.” *Journal of Heuristics* 8(1), 59–81.

Freling, R., R.M. Lentink, and M.A. Odijk. (2001). “Scheduling Train Crews: A Case Study for the Dutch Railways.” In S. Voß and J.R. Daduna (eds.), *Computer-Aided Scheduling of Public Transport,* pp. 153–165. Lecture Notes in Economics and Mathematical Systems, Vol. 505. Berlin: Springer.

Freling, R., R. Lentink, and A. Wagelmans. (2001). “A Decision Support System for Crew Planning in Passenger Transportation Using a Flexible Branch-and-Price Algorithm.” In S. Voß and J.R. Daduna (eds.), *Computer-Aided Scheduling of Public Transport,* pp. 73–90. Lecture Notes in Economics and Mathematical Systems, Vol. 505. Berlin: Springer.

Gamache, M. and F. Soumis. (1998). “A Method for Optimally Solving the Rostering Problem.” In G. Yu (ed.), *Operations Research in the Airline Industry*, pp. 124–157. Norwell, MA: Kluwer.

Gamache, F., F. Soumis, D. Villeneuve, J. Desrosiers, and E. Gélinas. (1998). “The Preferential Bidding System at Air Canada.” *Transportation Science* 32(3), 246–255.

Gamache, M., F. Soumis, G. Marquis, and J. Desrosiers. (1999). “A Column Generation Approach for Large Scale Aircrew Rostering Problems.” *Operations Research* 47(2), 247–262.

Gershkoff, I. (1989). “Optimizing Flight Crew Schedules.” *Interfaces* 19(4), 24–43.

Graves, G.W., R.D. McBride, I. Gershkoff, D. Anderson, and D. Mahidhara. (1993). “Flight Crew Scheduling.” *Management Science* 39(6), 736–745.

Hjorring, C.A. and J. Hansen. (1999). “Column Generation with a Rule Modelling Language for Airline Crew Pairing.” In *Proceedings of the 34th Annual Conference of the Operational Research Society of New Zealand*, Hamilton, New Zealand, December 10–11, pp. 133–142.

Hjorring, C.A., S.E. Karisch, and N. Kohl. (1999). “Carmen Systems’ Recent Advances in Crew Scheduling.” In *Proceedings of the 39th Annual AGIFORS Symposium*, New Orleans, USA, October 3–8, pp. 404–420.

Housos, E. and T. Elmroth. (1997). “Automatic Optimization of Subproblems in Scheduling Airline Crews.” *Interfaces* 27(5).

ILOG. (2002). *ILOG CPLEX User’s Manual.* Gentilly, France: ILOG.

Jarrah, A.I.Z. and J.T. Diamond. (1997). “The Problem of Generating Crew Bidlines.” *Interfaces* 27(4), 49–64.

Kohl, N. (1999). “The Use of Linear and Integer Programming in Airline Crew Scheduling.” In *Proceedings of the 3rd Scandinavian Workshop on Linear Programming*, Lyngby, Denmark, August 26–28.

Kohl, N. (2000). “Application of OR and CP Techniques in a Real World Crew Scheduling System.” In *Proceedings of CP-AI-OR’00: 2nd International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems,* Paderborn, Germany, March 8–10.

Kohl, N. and S.E. Karisch. (2000). “Integrating Operations Research and Constraint Programming Techniques in Crew Scheduling.” In *Proceedings of the 40th Annual AGIFORS Symposium,* Istanbul, Turkey, August 20–25.

König, H. and C. Strauss. (2000a). “Rostering-Integrated Services and Crew Efficiency.” *Information Technology and Tourism* 3(1), 27–39.

König, H. and C. Strauss. (2000b). “Supplements in Airline Cabin Service.” In D. Buhalis, D.R. Fesenmaier and S. Klein (eds.), *Information and Communication Technologies in Tourism 2000,* pp. 365–374. Berlin: Springer.

Lasry, A., D. McInnis, F. Soumis, J. Desrosiers, and M.M. Solomon. (2000). “Air Transat Uses ALTITUDE to Manage Its Aircraft Routing, Crew Pairing, and Work Assignment.” *Interfaces* 30(2), 41–53.

Lavoie, S., M. Minoux, and E. Odier. (1998). “A New Approach for Crew Pairing Problems by Column Generation with an Application to Air Transportation.” *European Journal of Operations Research* 35, 45–58.

Martins, E. and J. Santos. (2000). “A New Shortest Paths Ranking Algorithm.” *Investigacao Operacional* 20(1), 47–62.

ORTEC. (2003). `www.ortec.com`

Parrot. (1997). “Executive Summary.” ESPRIT 24.960.

SBS International. (2003). `www.sbsint.com`

Sellmann, M., K. Zervoudakis, P. Stamatopoulos, and T. Fahle. (2002). “Crew Assignment via Constraint Programming: Integrating Column Generation and Heuristic Tree Search.” *Annals of Operations Research* 115, 207–225.

Vance, P.H., C. Barnhart, E.L. Johnson, and G.L. Nemhauser. (1997). “Airline Crew Scheduling: A New Formulation and Decomposition Algorithm.” *Operations Research* 45(2), 188–200.