Επιχειρησιακή Έρευνα / Operational Research. An International Journal. Vol.5, No.2 (2005), pp.241-259

Solving the Airline Crew Recovery Problem by a Genetic Algorithm with Local Improvement

Yufeng Guo, Leena Suhl, Markus P. Thiel

Decision Support & OR Laboratory, and
International Graduate School *Dynamic Intelligent Systems*,
University of Paderborn,
Warburger Str. 100, D-33098 Paderborn, Germany
guo@dsor.de (Y. Guo), suhl@dsor.de (L. Suhl), thiel@dsor.de (M. P. Thiel)

Abstract

Within the complex and dynamic environment of the airline industry, any disturbance to normal operations has dramatic impact, and usually imposes high additional costs. Because of irregular events during day-to-day operations, airline crew schedules are rarely operated as planned in practice. Therefore, disrupted schedules should be recovered with as small changes as possible. In this article, we propose a genetic algorithm (GA) based approach, in which disrupted flights are reassigned within an evolutionary process. Because of the slow convergence rate achieved by conventional GA, a special local improvement procedure is applied in this approach. Computational results are reported for several disruption scenarios on real-life instances from a medium-sized European airline.

Keywords: airline crew recovery, airline crew rescheduling, airline crew scheduling, disruption management, genetic algorithm, meta-heuristic

1. Introduction

During the past decades, the airline crew scheduling problem (CSP) has been systematically studied by researchers in operations research, and significant results have been achieved by applying various algorithms and techniques. Basically, the major task of the CSP is to create a set of crew schedules with minimum operational cost. Every crew schedule consists of several sequences of flights and other types of activities, assigned to crews in a way that each flight is covered exactly (or at least) once by the required *crew complement* (the crew positions and the number of crew members required by a flight, e.g., a flight is usually assigned to one captain and one first officer for a short-haul cockpit crew problem).

Apart from the consideration of operational cost, labor regulations and agreements imposed by civil aviation authorities, union contracts, and company policies have to be fulfilled, and the workload should be evenly and fairly distributed among home bases and crew members [Suhl (1995); Kohl and Karisch (2004)]. More recently, due

to the increasing demand for a high quality-of-life level for crews, such problems become even more difficult to solve.

However, frequent disruptions, such as aircraft mechanical problems, severe weather conditions, crew unavailability, and air congestions, often imply high additional costs in today's complex operational environment. In practice, schedules are seldom operated exactly as planned; they are rather constantly disrupted by irregular events during day-to-day operations. Consequently, disturbances to normal operations change the planned schedule totally or at least partly. More importantly, tremendous costs have to be paid in order to recover from them. Throughout this article, we use the term crew recovery problem (CRP) for this type of problem.

Prior to the discussion of recent research on CRP in Section 2, we first refer some work on the CSP within past decades, because of the similarity between both problems. The CSP is typically divided into two sequential sub-problems [Barnhart et al. (1999)]: First, in the airline crew pairing problem (CPP) a set of pairings is generated that minimizes operational cost in such a way that each flight belongs to exactly one pairing. Second, the airline crew assignment problem (CAP) or airline crew rostering problem assigns generated pairings together with other pre-scheduled activities (trainings, vacations, and requested off-duty periods), which usually includes three practical approaches: *bidlines, personalized rostering* and *preferential bidding* [Kohl and Karisch (2004)]. Some researchers recently focus on integrated approaches combining the two steps into one [Guo et al. (2003)]. [Hoffman and Padberg (1993)] proposed a branch-and-cut method to solve the CSP optimally, while more recent approaches based on the branch-and-price technique show the capability to reduce the total solution time by tuning for specific problems [Barnhart et al. (1998)]. Our previous work on airline CSP was based on a so-called state-expanded aggregated time-space network flow approach [Mellouli (2001); Guo et al. (2003)]. In that approach, we developed the basic idea of applying states used by the vehicle maintenance routing problem to solve the airline CSP, and combined crew pairing and crew assignment in a partially integrated way that a certain interaction between these two steps is realized. Some heuristic approaches have been developed for the airline CSP as well, such as a GA-based approach for solving the airline CAP by [El Moudani et al. (2001)].

In this article, we propose a genetic algorithm based approach to solve the airline CRP. We introduce a novel two dimensional representation that is encoded with a matrix indicating the direct assignment of the flights. We first briefly describe the problem characteristics and the formulation in Section 2. It is followed by a detailed introduction to the genetic algorithm in Section 3. Furthermore, we introduce dedicated heuristics together with a local improvement procedure. Section 4 reports computational results tested on real-life instances from a medium-sized European airline.

2. The Crew Recovery Problem

In this section, we describe the airline crew recovery problem regarding its specific characteristics, and discuss a mathematical model for such problem.

2.1 Process and Problem Characteristics

As mentioned above, the process associated with the CRP takes care of disrupted situations in which original crew schedules require several or major modifications. When disruptions occur, a series of flights have to be delayed and even cancelled, and often additional crews and flights are required. The main task of rescheduling is to reassign disrupted flights as well and quickly as possible, and substantially assist airline coordinators in evaluating the updated plan.

Basically, three resources must be recovered in disrupted situations: aircraft, crew, and passengers. Each resource has a dramatic impact. For example, a shortage of aircraft may cause not only unexpected delays and cancellations, but also additional difficulties to the later crew rescheduling, i.e., crew may lose their connections or get stuck in an unfavorable airport. Usually, the complete recovery problem is usually decomposed into sub-problems, each is solved independently. The aircraft recovery is solved first to restore a flight schedule with respect to all maintenance requirements. The impact of disruptions upon passengers is reduced as much as possible by minimizing their inconvenience, and missing connections. Finally, crew has to be rescheduled under the updated situation. The way to decompose the entire recovery problem differs from airline to airline because of heterogeneous company rules. The reason of applying a sequential approach relies on the fact that a complete integration of the three phases is unrealistic from a practical point of view. However, better overall solutions may be achieved by allowing collaborations between the three steps of the sequential approach, and the first research results on this type of integrated disturbance measurement have appeared recently [Carmen Systems AB (2004)].

To our opinion, the work in solving the airline CRP is only at the beginning today. [Wei et al. (1997)] proposed an optimization model for handling disruptions, while [Yan and Lin (1997)] described the rescheduling problem caused by the closure of airports. Further systematic studies of the crew recovery problem were conducted by [Stojković et al. (1998)], in which they solve such a problem as an integer nonlinear multi-commodity network flow model with time windows and additional constraints. Dantzig-Wolfe decomposition combined with a branch-and-bound method is described in detail. Furthermore, [Lettovský et al. (2000)] proposed a pairing generation method working together with special branching strategies. [Yu et al. (2003)] described an award-winning real-life application employed by Continental Airlines in the US, in which the problem is treated as a set covering problem and a so-called generate-and-test heuristic is applied to generate rosters. Moreover, some

studies about airline irregular operations are also discussed by [Irrgang (1995)] and [Rosenberger et al. (2003)].

2.2 Problem Formulation

In the following, we assume that the aircraft recovery problem has been solved. In other words, there exists a modified flight schedule where some flights may be cancelled, delayed, rerouted or added. We define all these flights as "affected flights". The goal is now to provide an adequate crew for each (affected or not affected) flight, so that the updated crew rosters, with minimal variation from previously planned rosters, produce as little additional cost as possible. In addition, we consider different crew positions separately, i.e. captain or first officer, as it may reduce the size of the problem without influencing the quality of the final result.

Similar to the airline CSP, the airline CRP can be mathematically formulated as a set partitioning type model, where a set of affected flights caused by disruptions needs to be assigned or reassigned exactly once. These disrupted flights grouped with previously planned flights are chained into a huge amount of rosters, which represent all possible individual schedules for crew members within the recovery period (an interval of time usually started from the earliest disruption, and ended within the day or some days later, within which the rescheduling is carried out). Each crew member, therefore, will be finally assigned at most one revised schedule for the examined time period with respect to all regulations and rules.

In this approach, we apply the concept of integration, i.e., the problem is solved in an integrated way instead of addressing pairing generation prior to the assignment phase. Rosters for individual crew members are generated directly from the flight leg (a non-stop flight with its crew complement and fleet requirements) level. This is only possible because the recovery period is normally much shorter than the period examined in the planning phase. The problem is treated as a set partitioning model, where a set of rosters is given and needs to be assigned to a certain number of individual crew members, by which all the flights are covered exactly once. We start with the following definitions prior to the complete model:

F, set of flights

R, set of possible rosters

W, set of crew members

c_i^w, operational cost of assigning the roster i to the crew member w

u_f, additional cost, if the flight f is assigned to a standby or reserve crew

v_i^w, the penalty to changes (variation) from originally planned schedule

$a_{fi} = 1$, if flight f is included in the roster i, 0 otherwise

$b_{iw} = 1$, if roster i belongs to crew member w, 0 otherwise

The binary decision variables are:

$x_i^w = 1$, if roster i is assigned to crew member w, 0 otherwise

$y_f = 1$, if flight f is not assigned to any crew member in service, 0 otherwise

Therefore, the model can be expressed as:

$$\textbf{min.} \sum_{w \in W} \sum_{i \in R} (c_i^w + v_i^w) x_i^w + \sum_{f \in F} u_f y_f \tag{1}$$

$$\textbf{s.t.} \sum_{w \in W} \sum_{i \in R} a_{fi} x_i^w + y_f = 1 \quad \forall f \in F \tag{2}$$

$$\sum_{i \in R} b_{iw} x_i^w \leq 1 \quad \forall w \in W \tag{3}$$

$$x_i^w \in \{0,1\} \quad \forall i \in R,\ w \in W$$

$$y_f \in \{0,1\} \quad \forall f \in F$$

where the first part of the objective function (1) denotes minimizing the total operational cost c_i^w, together with the disturbances to the crew v_i^w realized by expressing the changes in a monetary sense. v_i^w equals zero, if the corresponding roster is identical with an original roster. Details of their calculation are given in Section 3.5. Those flights which can not be assigned to any crew member in service, requiring reserve and standby crews, which imposes additional costs u_f. Constraint (2) guarantees that all flights ($f \in F$) are covered exactly once, while constraint (3) ensures that each crew member ($w \in W$) takes at most one roster ($i \in R$).

Some preliminary work was done to solve the above model with several instances. However, the results indicate that only relatively small real-life problems are suitable to be solved directly by standard integer optimizers, such as ILOG CPLEX [ILOG (2002)] and MOPS [Suhl (1994)]. Medium- and large-sized instances require too much computational time that is apparently impractical in reality, especially in the case that the recovery period is comparably long, e.g., for large instances it often takes more than 30 minutes to be solved.

Due to the extremely large number of possible rosters, common methods apply efficient roster generation procedures to build a model gradually. A number of approaches follow the idea of column generation to implicitly construct the model. Consequently, the total solution time can be significantly reduced, since only a small set of possible rosters is used. For further details we refer to [Barnhart et al. (1998)] and [Fahle et al. (2002)]. In the following sections, we will fully describe a GA-based approach and report our computational results.

3. A Genetic Algorithm Based Solution Method

Within the past years, genetic algorithms were successfully applied to many applications, as they can gradually find better solutions during the course of an evolution. A lot of research has been carried out after the basic concept of genetic algorithm was introduced by [Holland (1975)]. For example, as a combinatorial problem, the classical set covering problem (SCP) was systematically examined by [Beasley and Chu (1996)], who observed the potential for solving such type of problems.

As an analogy to the theory of evolution in biology, a genetic algorithm basically works with a group of candidate solutions (*individuals*), called *population*. Each individual in the population is encoded into a specific representation with regard to the problem. The new generation (*offspring*) is produced from one or more individuals (*parents*) by applying recombination methods called variation operators, e.g., crossover and mutation. Every individual is measured and attached a value (*fitness value*) showing how "good" it is. The selection of parents and the survival of offspring may be determined randomly or based on their fitness value. In this way, the convergence to an acceptable or optimal solution is accomplished.

In our approach, we customize the conventional genetic algorithm into a hybrid genetic algorithm. It includes a set of heuristics with the knowledge of this specific problem domain, together with a so-called *local improvement* procedure acting as a supplementary local search to the genetic algorithm. Details of the approach, such as the coding scheme, special variation operators, and the fitness function are described in the following subsections.

3.1 Chromosome Encoding

The most natural representation of a chromosome is probably a one-dimensional string, such as the ordered city list for the traveling salesman problem [Michalewicz and Fogel (2000), chap. 7] and the activity permutation list for the resource-constrained project scheduling [Hartmann (1998)]. However, according to our experience, string representation is not suitable to solve the airline CSP and CRP because of the complex structure of their solutions. We propose a two-dimensional matrix to represent a direct assignment of flight legs to crew members. Part of the

chromosome can be seen in Fig. 1. Each column indicates a certain time slot (each time slot represents one hour period), within which a number of flights depart. Each row stands for a roster for a specific crew member, i.e., the number of rows is equal to the number of crew members. All items in the cells of the matrix represent the sequential numbers of the flight legs. Pre-scheduled activities, such as vacancies, training and requested off-duty etc., are represented as flight legs with an additional characteristic that they are fixed and cannot be changed and/or removed from the given cell (marked with asterisk, e.g., flight leg F1 in the first column).

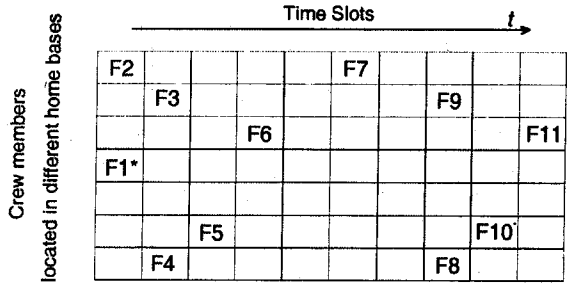

Figure 1. *Two-dimensional representation*

Changes to the crew schedule in such a representation underlie the following assumptions:

- Flight legs can only be moved between rows but cannot be moved to different columns as the departure times are fixed to exactly one time slot (column)
- One cell in the matrix can only contain at most one flight leg, because no pair of two sequential flight legs depart within the same time slot. (it is impossible to service two flights in less than one hour)
- A flight leg can only be assigned to one crew member, as only one crew position is involved at a time, i.e., each flight leg is unique within one matrix.

3.2 Initialization

Initialization is a process to generate the initial population which allows the application of variation operators. A certain number (*population size*) of individuals, exhibiting equal or similar genome structures, is created either randomly or heuristically. Generally speaking, seeding with a randomly generated solution in the initial population comes along with wide diversity. However, some researchers have reported that a population with a higher quality starting solutions obtained from dedicated procedures can help the algorithm perform faster and find better solutions [Reeves (1993), chap. 4], but the risk of premature convergence also increases.

Apart from the random initialization of the population, two other strategies are applied to generate initial individuals: The first initialization heuristic checks the

arrival time of the previous flight leg, once a flight leg has been assigned. If the flight leg does not fit in a specific position of the matrix, another crew member needs to be evaluated. If no further row is found, the flight leg is assigned randomly. By using this heuristic, the individuals start with a higher fitness value, although most of them may be still infeasible. However, the drawback is that individuals sometimes get very similar and the risk of ending up in local optima increases.

One part of the objective function is to minimize the changes to the original schedule, i.e., parts of the final solution stay unchanged. Therefore, we take the advantage of such characteristics of the problem, a procedure, *evolutive initialization*, is implemented. It generates a certain number of individuals by keeping unaffected flights $F_{unaff}(F_{unaff} \subset F)$ intact and assigning the rest to crew members who are able to provide the service (see Fig. 2). In other words, most crew members do not change their schedules, if they are not directly influenced by current disruptions.

```
Procedure evolutive initialization
begin
  remove_affected_flights (original_schedule)
  foreach f
    W = feasible_crewmember_set (f)
    w = select_crewmember_randomly (W)
    assign (f, w)
  endfor
end
```

Figure 2. *The procedure of evolutive initialization*

According to our experience, in order to apply an effective genetic algorithm, an initial population with a high diversification is very useful. This can be achieved by combining the three methods described above, i.e., random, heuristic and evolutive initialization procedure.

3.3 Variation Operations

For the string representation based genetic algorithm, two commonly applied operators are two-point crossover and single bit mutation. However, specific operators have to be introduced for this real-life complex problem structure. In this paper, two main categories of variation operators were applied. The crossover operators select two individuals as parents and recombine them; mutation operators are applied on only one individual.

3.3.1 Crossover Operators

We started with implementing a simple crossover operator, *row-based crossover*, in which a given range of rows of one parent are replaced by corresponding rows from the other parent. However, as a consequence a flight leg is often served by more than

one crew member in the offspring, or a flight leg is not included at all. Therefore, we propose a correction procedure to deal with both effects by inserting missing flight legs, or deleting doubled flight legs randomly.

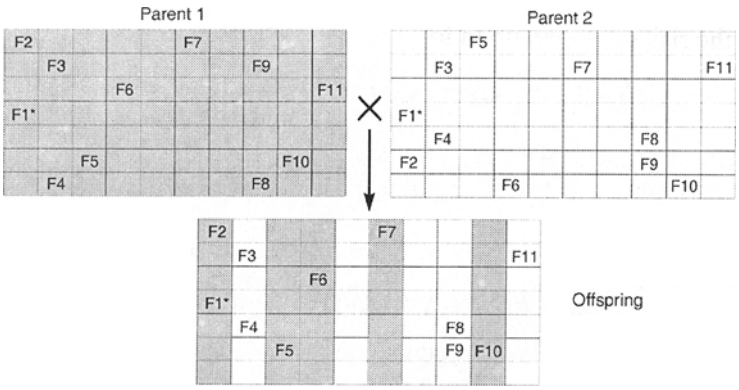

Figure 3. *Column-based crossover operator*

In contrast to the row-based crossover, a *column-based crossover* was designed. Its basic idea is to construct new offspring in a way that each column is selected randomly or heuristically from the parents chosen. As illustrated in Fig. 3, in order to maintain the feasibility of the offspring, this special crossover tries to preserve "good" ranges of the columns containing valid pairings. Feasibilities are preserved, when two consecutive flight legs are connected at the same airport, or there is sufficient time between the two flight legs. From the illustration, one may note that the sequence of F9 and F10 can express infeasibility, because the departure time of F10 is earlier than the arrival time of F9, or the ground time between the two flight is not adequate.

3.3.2 Mutation Operators

The intention of mutation operators is to avoid getting trapped in local optima through increasing the diversity of the population. We introduce two mutation operators which have been applied in our approach.

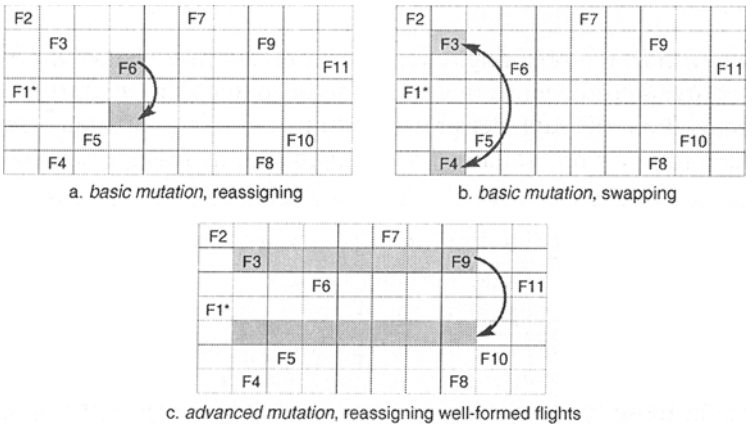

Figure 4. *Mutation operators*

First, a *basic mutation* chooses randomly one flight leg from a chromosome to be mutated and one crew member in the crew member set F. If the position of the chosen flight leg is empty, it is moved to that position (see Fig. 4.a). If not, these two flight legs are swapped (see Fig. 4.b). Nothing is changed, if the new position is exactly the same to the current position.

The second mutation operator, *advanced mutation*, includes a heuristic procedure that substitutes the random selection of the new crew member described above. After a flight leg is randomly chosen, a new crew member (one row in the matrix) is selected and evaluated according to certain restrictions: We take into account whether the possible transition and subsequent flight legs are suitable for the inserted flight leg in terms of checking the departure and arrival airport and the underlying times. The corresponding cost is calculated for assigning the flight leg which includes the cost for flight duty time, maximum duty period time, hotel stays and rest periods (more details can be found in Subsection 3.5). The crew member with the lowest cost is chosen, and the flight leg is then assigned to that crew member. Similarly, if some flights already occupy the position examined, then these two flight legs are swapped. Furthermore, a complete sequence of flights is moved to the new crew member, if a sequence of flights is recognized as "well-formed" (normally one day work for a crew member, see Fig. 4.c as an example). Such a sequence of flights can be identified by checking arrival/departure airports, as well as arrival/departure times of the flights.

3.4 Local Improvement

Although the traditional genetic algorithm works well on some problems, hybrid genetic algorithms, especially in combination with algorithms reflecting specific knowledge of the problem, perform mostly better.

As operators presented above might produce an infeasible solution, a dedicated local search procedure, *local improvement*, is applied under a specified probability directly after the generation of new offspring. Strictly speaking, the purpose of such a procedure is to improve one solution by finding one of its feasible neighbor solutions.

Basically, this procedure goes through all infeasible assignments in the solution, and repairs them by reassigning the flights to other crew members. Flights are assigned to randomly selected crew members, while feasibility is maintained by adding a filter to select only possible crew members, or by swapping flights which will possibly improve the current solution. The reason why we choose such a heuristic rather than other well-known modern heuristics (e.g., simulated annealing etc.), is that the running time for those type of local searches is usually not acceptable in practice.

3.5 Evaluation

According to the nature of the evolution, one has to find out which individuals may survive in what sense of the measurement. A proper evaluation function is the right way to do so. The evaluation procedure is applied to each individual, determining its quality compared to the whole population. In this approach, *real cost*, *virtual cost* and *change cost* are applied to the individual as the objective is to minimize the occurring cost calculated by the objective function.

Real costs C_{opl} (operational costs) are the costs that appear in the algorithm and in reality if the computed solution is applied in the final schedule. They consist of hotel cost that arises when a crew member takes an overnight rest at another airport rather than his or her home base, and the proceeding cost occurring when a crew member is transited from one airport to another by means of taxi or train.

Virtual costs C_{inf} have been implemented as a penalty for rosters that do not comply with restrictions. In the case that it is impossible to create a feasible connection between two flight legs, a high penalty is introduced. Likewise, certain penalty is imposed, if the time between two flight legs is too short for check-out and check-in, or the next flight starts even earlier than the arrival time of the previous flight.

In addition to these main restrictions, there are other rules and regulations which the algorithm has to take into account, defined by the airline, collective labor agreements and other regulation committees such as governmental instances. For example, the following regulations cause virtual cost if one or more of the restrictions is not met:

- Maximum daily/weekly/monthly flight hours
- Minimum rest time between flight duties
- Maximum sequential working days and minimum weekly rest days
- Weekly rest at crew member's home base

Change cost C_{chg} is the sum of change penalty v_i^w of all the rosters in the solution. It is calculated in such a way that a certain amount of penalty is set to each change depending on how much the flight is involved in the situation and which category the change belongs to. It can be calculated as $C_{chg} = \sum_{f \in F} C_{chg}^f$, where

$$C_{chg}^f = \begin{cases} 0 & \textbf{if the change is in the disrupted period, and the crew member is originally affected;} \\ P1 & \textbf{if the change is in the disrupted period, and the crew member is not originally affected;} \\ (\frac{d}{D})^{s1} \cdot P2 & \textbf{otherwise} \end{cases}$$

A constant penalty $P1$ is imposed if the change occurs within the disrupted period (the period starts from the departure time of the earliest affected flight to the arrival time of the latest one) and the corresponding crew member is not originally affected by disruptions. In other words, there is no direct need for this specific crew member to change his/her original schedule due to disruptions. A higher penalty is given in the case that the change occurs earlier or later to the disrupted period. While d is the number of days away from disrupted period; D is the total examined time period in days; $P2$ is the penalty for those changes. The exponent $s1$ is set to 1 in this approach.

The evaluation is an adaptive function that depends on the stage of the evolution process. In the early stage, most of the individuals are infeasible, therefore, the quality measurement is realized by considering the first two costs, real and virtual cost. After a certain number of generations or reaching a given percentage (e.g., 80% in our approach) of feasible solutions in the population, the operational cost and change cost become the major part of the evaluation.

3.6 Selection and Replacement Strategies

To guarantee an appropriate selection and reproduction according to the fitness value of an individual, a dedicated probability is calculated. Since the variance between fitness values can be quite large, the risk that some individuals dominate the whole population after a few generations must be taken into account. Hence, ranking-based selection strategies rather than strategies based on the absolute fitness value seem to be more applicable for this specific problem. For more discussion we refer to [Michalewicz and Fogel (2000)] and [Thierens and Goldberg (1994)].

[Bölte and Thonemann (1996)] discussed a selection scheme which operates based on the ranking of individuals. However, in order to differentiate individuals more precisely, a revised formula is used in this approach. In (4), let $R(i)$ be the rank of an individual i according to its fitness value and n be the total number of individuals in the population. Then, the probability $P(i)$ that individual i is selected for reproduction is calculated with the following formula:

$$P(i) = \frac{R(i)^s}{\sum_{j=1}^{n}(R(j)^s)} \tag{4}$$

With parameter s, it is possible to control the probabilities according to the ranks more specifically. If $s > 1$, the differences between two probabilities of individuals increases, so that better individuals have not only a higher but an exponentially higher chance to be selected.

In comparison to [Holland (1975)], the population in this approach is not replaced by the new offspring as a whole, and the best individual is always preserved during each iteration. A certain percentage of individuals in the population is replaced by their offspring. In our approach a value around 80% is usually adopted in the experiments.

4. Computational Results

Several experiments were conducted based on the real-life practical setting of a medium-sized European tourist airline. The network is a point-to-point type of network with several home bases (in contrast to the hub-and-spoke structure often observed in the US). Within such a network, multiple home bases are located in Germany, while many other airports are mainly spread out around Europe. The input data are the real life flight schedules of one fleet together with the information regarding the crew members' availability and their home bases and destination airports. An overview of the seven tested instances is shown in Table 1.

Before elaborating on the details of the observed results, a brief introduction to the testing scenarios is given (see Table 2). We classify all disruptions into three main categories by their scale of severity, namely, minor, medium, and major disruptions. This can be understood as the total number of affected flights together with those subsequent flights which are influenced by them, indicating how many flights have to be rescheduled at least. In addition, the number of affected home bases and crew members as well as the length of the recovery period are further crucial factors for the difficulty of the instances.

Table 1. *Problem instances*

Instance	Home bases	Hotel bases	Airports	Flights	Crew members	Duration (day)
A1	6	8	52	159	47	15
A2	5	10	57	228	42	15
A3	6	13	66	268	42	10
A4	6	14	65	275	42	10
A5	6	14	66	406	42	15
A6	6	16	69	415	42	15
B1	11	21	66	1287	188	10

The approach described in this article has been implemented in ANSI C++ language, and numerous tests were conducted on a regular PC with Intel Pentium IV 2.2 GHz CPU, 2 GB main memory, running Microsoft Windows XP Professional operating system.

Table 2. *Disruption scenarios*

Category	Scenario	Affected flights	Aggregated affected flights	Affected home bases	Affected crew members	Recovery period (hours)
Minor	A1S1	1	1	1	1	15
	A2S2	1	2	1	1	12
	A3S3	2	3	1	2	16
	A4S4	2	2	2	2	20
Medium	A5S5	4	5	1	4	24
	A5S6	4	5	1	4	36
	A6S7	4	3	2	4	40
Major	A5S8	5	7	1	5	72
	B1S9	13	15	1	13	111

After a number of tests, the results show that the best parameter setting for one instance does not perform the same in all the other instances, e.g., the number of generations for solving a small instance is relatively lower than that for large instances. Furthermore, a larger population size normally produces better results (see Fig. 5, the test is based on instance A5), the best size, however, varies from instance to instance. Based on the results of the experiments, a population size of 25 shows more efficiency for most medium-sized and large-sized of instances, while a setting with 45 or more performs better for small instances.

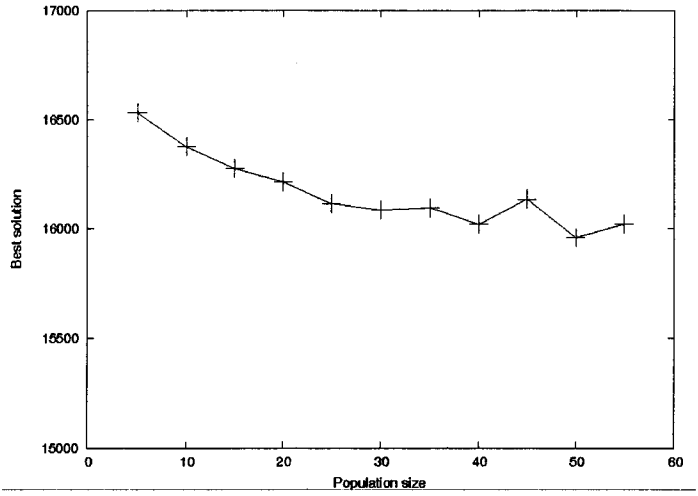

Figure 5. *Effect of the population size*

The local improvement procedure usually produces faster convergence and better results. The comparison indicating the difference between using and not using the local improvement procedure can be seen in Fig. 6. In this example, we take the scenario A6S7, and the improvement procedure starts after 300 generations. As one can see, the best individual is always better than the one found by the algorithm without the local improvement after 300 generations. Because the local improvement method maintains feasibility of new offspring, the speed of improving solutions, therefore, can be much faster.

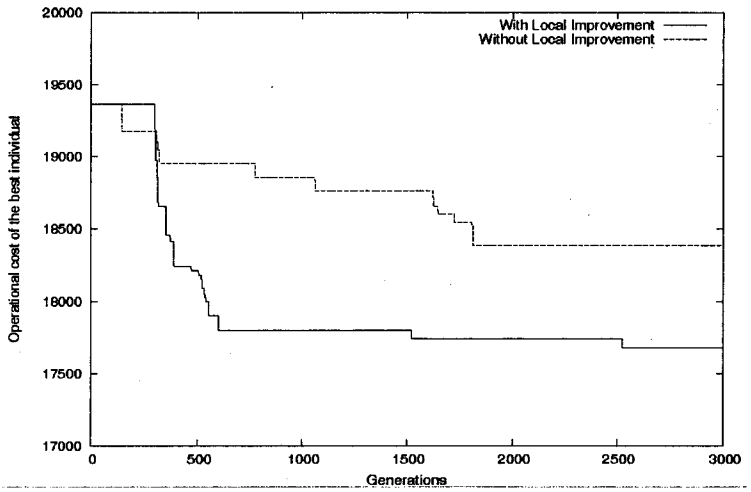

Figure 6. *Comparison between using and not using local improvement*

Table 3. *The performance of different crossover operators*

Instance & scenario	Row-based crossover		Column-based crossover		Mixed crossover	
	Cost *	Time **	Cost	Time	Cost	Time
A1S1	1300	9.6	1300	9.6	1300	9.6
A2S2	6950	12.9	6780	13.2	6880	12.8
A3S3	8260	14.2	8260	14.7	8260	14.2
A4S4	11000	14.3	11590	14.5	11320	14.5
A5S5	16390	21.8	16165	22.3	16125	22.3
A5S6	15750	22.4	15890	22.6	15870	22.3
A6S7	18635	22.7	18195	22.7	18430	23.3
A5S8	15965	23.2	16470	23.2	15315	29.9
B1S9	26290	372.8	25945	363.2	26530	482.8

* Operational cost. ** Computational time for each generation in millisecond

The result produced by the column-based crossover operator is slightly better than that produced by the row-based crossover. Interestingly, the combination of these two operators with a certain probability (0.5 used in the example) sometimes performs even better than one crossover alone, but it needs mostly more computational time. The best solutions found by using the two different crossovers and the combination of both are listed in Table 3. In Fig. 7, the general performance on instances A5S5 and B1S9 is presented, in which the column-based crossover in both examples produces faster convergence and better final solutions.

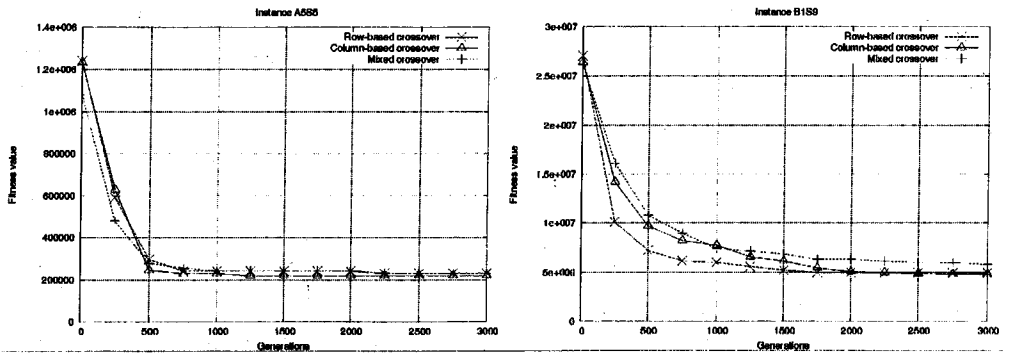

Figure 7. *The performance of crossover operators*

Regarding the quality of the final solution, the algorithm can find an optimal solution rather fast for small-sized instances, but it turns out to be difficult to find an optimal solution for larger problems, e.g., A6 and B1. This is true especially when the recovery period is comparably long, i.e., the total number of involved flights is large.

However, the solutions finally found by this algorithm are all reasonable in terms of the minimization of operational cost and the variation from the original crew schedule. The results also show that the final solution, for most instances with minor or medium disruptions, are feasible, and do not produce any further operational cost. In addition, for most cases only a limited number of notifications is required, which is acceptable in practice. Most importantly, the overall solution time required, usually a few minutes at most, is much shorter than what a column generation-based approach needs, and reasonable solutions can be found. It is particularly useful in the situations under time pressure.

5. Conclusions

In this article, we present a GA-based heuristic which aims to solve the difficult real-life airline CRP for unexpected operational situations caused by disruptions. We propose a new structure for the chromosome representation. Accordingly, several variation operators were implemented, and the paper shows their capability of finding "better" solutions step by step.

With the help of our previous experience in studying the airline CSP, various heuristics we applied show the efficiency in a way that these ad hoc methods aim to account for specific characteristics of the problem. And the proposed solution strategy in this paper has proven to be effective. This was only achieved because several dedicated heuristics described above have been used to support the exploration of the solution space. Nevertheless, the genetic algorithm has to be customized for each problem instance that is solved, especially for large instances. Therefore, deeper investigations on the examined instances need to be conducted in the future, so that the combination of several cooperative fine-tuned operators and more reasonable parameter settings will perform even more efficiently.

References

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. Operations Research 46, 316–329.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Vance, P. H. (1999). Crew scheduling. In: Hall, R. (Ed.), Handbook of Transportation Science. Kluwer Academic Publisher, Norwell, pp. 493–521.

Beasley, J., Chu, P. (1996). A genetic algorithm for the set covering problem. European Journal of Operational Research 94, 392–404.

Bölte, A., Thonemann, U. W. (1996). Optimizing simulated annealing schedules with genetic programming. European Journal of Operational Research 92 (2), 402–416.

Carmen Systems AB (2004). Carmen integrated operations control. URL http://www.carmen.se/

El Moudani, W., Cosenza, C., de Coligny, M., Mora-Camino, F. (2001). A bicriterion approach for the airline crew rostering problem. In: Zitzler, E., Deb, K., Thiele, L., Coello, C. A. C., Corne, D. (Eds.), Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001). Springer-Verlag, Berlin, pp. 486–500.

Fahle, T., Junker, U., Karisch, S., Kohl, N., Sellmann, M., Vaaben, B. (2002). Constraint programming based column generation for crew assignment. Journal of Heuristics 8, 59–81.

Guo, Y., Mellouli, T., Suhl, L., Thiel, M. P. (2003). A partially integrated airline crew scheduling approach with time-dependent crew capacities and multiple home bases. Tech. rep. WP0303, DS&OR Lab., University of Paderborn, Germany, accepted by European Journal of Operational Research.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 45, 733–750.

Hoffman, K. L., Padberg, M. (1993). Solving airline crew scheduling problems by branch-and-cut. Management Science 39 (6), 657–682.

Holland, J. (1975). Adaption in natural and artificial systems. The University of Michigan Press.

ILOG (2002). Cplex v8.0 User's Manual. ILOG, France.

Irrgang, M. E. (1995). Airline irregular operations. In: Jenkins, D., Ray, C. (Eds.), Handbook of airline economics, 1st Edition. McGrawHill Aviation Week Group, New York, pp. 349–365.

Kohl, N. and Karisch, S. E. (2004). Airline crew rostering: Problem types, modeling, and optimization. Annals of Operations Research, 127:223–257.

Lettovský, L., Johnson, E. L., Nemhauser, G. L. (2000). Airline crew recovery. Transportation Science 34 (4), 337–348.

Mellouli, T. (2001). A network flow approach to crew scheduling based on an analogy to a train/aircraft maintenance routing problem. In: Voß, S., Daduna, J. (Eds.), Computer-Aided Scheduling of Public Transport. Springer, Berlin, pp. 91–120.

Michalewicz, Z., Fogel, D. B. (2000). How to Solve It: Modern Heuristics. Springer-Verlag.

Reeves, C. R. (1993). Modern heuristic techniques for combinatorial problems. John Wiley & Sons.

Rosenberger, J. M., Johnson, E. L., Nemhauser, G. L. (2003). Rerouting aircraft for airline recovery. Transportation Science 37 (4), 408–421.

Stojković, M., Soumis, F., Desrosiers, J. (1998). The operational airline crew scheduling problem. Transportation Science 32 (3), 232–245.

Suhl, L. (1995). Computer-Aided Scheduling—An Airline Perspective. Deutscher Universitäts-Verlag (DUV), Wiesbaden.

Suhl, U. H. (1994). MOPS: A Mathematical OPtimization System. European Journal of Operational Research 72, 312–322.

Thierens D. and Goldberg D. E. (1994). Convergence models of genetic algorithm selection schemes. In PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature, pages 119–129, London, UK, Springer-Verlag.

Wei, G., Yu, G., Song, M. (1997). Optimization model and algorithm for crew management during airline irregular operations. Journal of Combinatorial Optimization 1 (3), 305–321.

Yan, S., Lin, C. G. (1997). Airline scheduling for the temporary closure of airports. Transportation Science 31 (1), 72–83.

Yu, G., Argüello, M., Song, G., McCowan, S. M., White, A. (2003). A new era for crew recovery at continental airlines. Interfaces 33 (1), 5–22.