Using Heuristics to Solve the Dedicated Aircraft Recovery Problem

Michael Løve * Kim Riis Sørensen † Jesper Larsen ‡
Jens Clausen §

Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Kongens Lyngby, Denmark

November 12, 2001

Abstract

The Dedicated Aircraft Recovery Problem (DARP) involves decisions concerning aircraft to flight assignments in situations where unforeseen events have disrupted the existing flight schedule, e.g. bad weather causing flight delays. The dedicated aircraft recovery problem aims to recover these flight schedules through a series of reassignments of aircraft to flights, delaying of flights and cancellations of flights.

This article describes an effective method to solve DARP. A heuristic is implemented, which is able to generate feasible revised flight schedules of good quality in less than 10 seconds when applied to real flight schedules with disruptions from British Airways. The heuristic is able to consider delays, cancellations and reassignments simultaneously and balance the trade-off between these options. It is also demonstrated that different strategies can be applied to prioritize these options when generating the revised flight schedules without affecting the solution time required.

Keywords: Disruption Management, Optimization, Airline Planning, Dedicated Aircraft Recovery, Heuristic, Irregular Operations

*Email: michael@loeve.info. FDB (Logistic Development), Albertslund, Denmark
†Email: kim@carmen.se. Carmen Consulting, Copenhagen, Denmark
‡Email: jla@imm.dtu.dk.
§Email: jc@imm.dtu.dk.

1 Introduction

The Dedicated Aircraft Recovery Problem arises when unforeseen events have disrupted an existing flight schedule, e.g. bad weather causing flights to be delayed. In such a situation, the main goal for the airline is to restore or recover the flight schedule as much as possible, i.e. minimize the number of cancellations and the total delay. Similar recovery problems exist in other areas, for a introduction to Disruption Management see [CHLL01].

The work described here has been carried out partly in the DESCARTES project, a project funded by the European Union, the Technical University of Denmark, British Airways and Carmen Systems (see [DLT01]). The Dedicated Aircraft Recovery Problem (DARP) is the term used in this project to describe the problem briefly described above. The word *dedicated* refers to the fact that a single resource is considered, namely aircraft.

DARP has been given other names by different researchers and its precise definition varies accordingly. Examples include Flight Operations Decision Problem (FODP), the Operational Daily Airline Scheduling Problem (ODASP), the Daily Aircraft Routing and Scheduling Problem (DARSP), Airline Operations Control Center Problem (AOCCP) and Airline Schedule Perturbation Problem (ASPP). However, all these names basically refer to the same problem or aspects thereof.

Because of the differences in definition, a precise definition of DARP as considered in this article is important: Given an original flight schedule and one or more disruptions, the Dedicated Aircraft Recovery Problem consists of changing the flight assignments of the aircrafts in order to produce a feasible and more preferable revised flight schedule. Changes can be delaying flights, cancelling flights, swapping aircraft (either within the same fleet or between fleets) or use of standby aircraft.

The *flight schedule* includes all flights flown within a certain period of time by a given fleet including the original departure and arrival times, the expected flight durations and the *tail assignments*. Tail assignments refer to specific aircraft being assigned to specific flights. The term *swap* means that two flights, originally designated to be undertaken by two specific aircraft, are interchanged between these aircraft. This is illustrated later in Figure 2.

1.1 Decision Costs

Central in DARP are the decision costs. These costs must reflect how *preferable* each decision possibility is. Feasible solutions are compared by cost, and thus the ability to quantify the quality of a solution is absolutely necessary.

Generally, the most common approach to quantifying decision costs is to estimate the *real* costs associated with each decision possibility. As mentioned, 3 decision

possibilities exist when solving DARP, namely delaying, cancelling and swapping. If the actual costs of these decision possibilities are calculated, they would have to include such factors as ill-will from customers, costs of customers missing down-line flights, crew planning issues, etc. These costs and many others are important to include, yet most of them are very difficult, if not impossible to estimate.

It may therefore be futile to base an objective function on **real-world** operating costs when solving DARP. However, when viewing a computerized decision support tool for recovery as a tool generating good feasible options the real-world cost of a particular solution is not necessary in the generation phase, where the costs are used to guide the search for good solutions. In addition, costs like ill-will are in addition to the quantification itself also difficult to evaluate empirically since such an evaluation requires contact with all disrupted passengers after the disrupted flight. We have therefore instead concentrated on quantifying the preferability of the basic operating principles that flight controllers adhere to.

In general, these principles are to minimize the number of cancellations, the total delay and to make as few swaps as possible. However, as will be demonstrated, there is a clear trade-off between these 3 objectives and the quality of a solution to DARP is ultimately a matter of preference. By focusing on the basic operating principles, different solution strategies are easily applied that can accommodate these preferences. One example is the trade-off between the number of swaps and the total delay: Depending on the situation at hand, the number of swaps that are acceptable varies. A strategy can thus be applied that attempts to recover the flight schedule by making a minimum number of swaps while accepting a number of delays. More examples are given in Section 7, where our heuristic is tested on British Airways flight schedules.

2 Previous Work on DARP

A number of authors have worked on problems similar to DARP. Teodorović and Gubernić introduce a method of solving DARP that focuses on minimizing the total passenger delay in [TG84]. Later Teodorović together with Stojković introduces a different model that aims to minimize the number of cancellations and the total passenger delay. They attempt to do so by using lexicographic optimization and present their work in [TS90]. In [JYKR93] Jarrah et al. introduce two separate models that minimize delays and cancellations respectively. The models thus are not able to consider the trade-off between cancelling and delaying. Yan and Yang formulate a model in [YY96] that aims to minimise the period of time in which the flight schedule is disrupted. In [CK97] Cao and Kanafani introduces the first model which seemingly can consider delays and cancellations simultaneously while solving problems of a realistic size. However, as it is demonstrated in [LS01b], this model contains several errors. Among other problems, the model does not prevent non-existent aircraft from undertaking flights and in some instances, the model

cannot associate correct costs with a decision possibility. Lastly, Argüello et al. demonstrate a heuristic approach to solving DARP in [ABY98] and in [ABY97]. None of the methods described briefly are able to solve realistic examples of DARP effectively.

3 Motivation for Using Heuristics to Solve DARP

Dedicated aircraft recovery occurs every day in most airlines. It is done manually with the assistance of various tools such as computer based graphical interfaces that allow controllers easy access to information on the present situation. Among controllers there is an outspoken wish for decision-support tools, tools that give good solutions quickly. Such tools may in this case be built using methods based on mathematical modelling to solve the DARP. However, there are a series of difficult criteria to meet.

Firstly, a decision support tool to solve DARP has to be able to handle problems of a realistic size, e.g. 50 airports, 100 aircraft and 500 flights. Secondly, a tool preferably has to produce a result in less than 3 minutes. And finally, the must be flexible, i.e. able to accommodate for new or modified restrictions frequently.

At the same time DARP is a very complex problem: The cost of a certain decision depends on other decisions made earlier, e.g. the ready-time of an aircraft at a given airport depends on whether the aircraft was delayed or reassigned earlier that working day. This dependency along with a large solution space and integral decision variables makes DARP a very difficult problem to solve.

Given all these factors, heuristics seem an obvious choice. Flight planners are mainly interested in a good solution to DARP quickly and heuristics are often able to meet this criteria. Similarly, new or modified restrictions are often relatively easy to implement when using heuristics as opposed to algebraic representations of DARP.

4 Basic Heuristic Design

4.1 Definition of Network

As basis for the design of a DARP-heuristic, a network representation of the data and problem is chosen. This representation is illustrated in Figure 1. The vertical axis depicts the time of day. At each airport, the nodes on the left are called aircraft nodes, because they represent aircraft. These nodes are placed at the point in time when the aircraft is ready to depart. The nodes immediately to the right of the aircraft nodes are flight nodes and they represent scheduled departures of flights. Likewise, these are placed at the point in time when the flight is scheduled to depart.

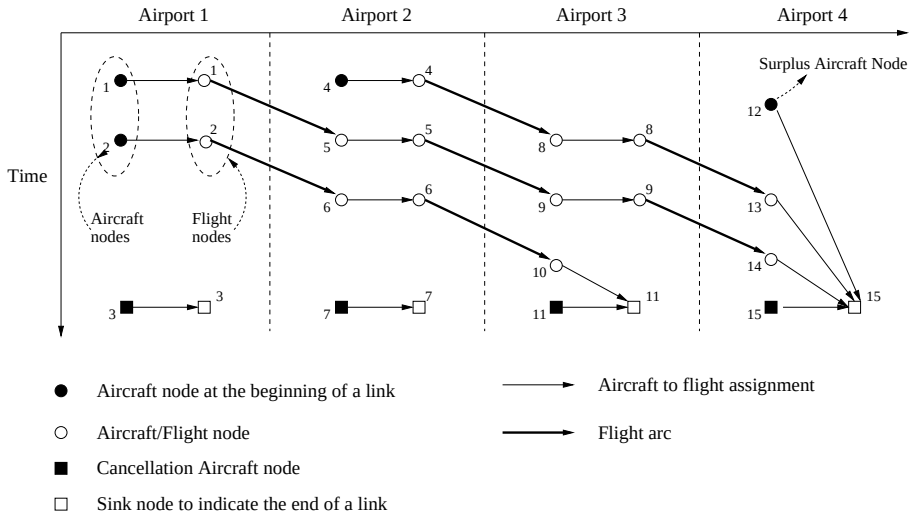


Figure 1. *Underlying network to use with a DARP-heuristic.*

The arcs connecting the aircraft and flight nodes are essential to the representation of the flight schedule. Such an arc represents that the particular aircraft is assigned to undertake the flight to which it is connected. The heuristic modifies the flight schedule by altering these assignments through *swaps*. A swap is illustrated in figure 2.

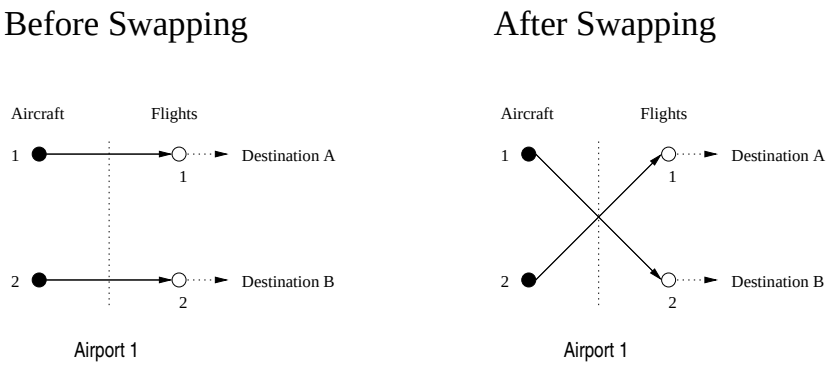


Figure 2. *A swap.*

The network in Figure 1 is based on the network introduced in [CK97], however, there are significant differences. The term *flight link* is used to describe a sequence of flights performed by one aircraft. In Figure 1 an example of a flight link is the sequence $2 \to 2 \to 6 \to 6 \to 10 \to 11$. In [CK97] it is assumed that a time horizon of 3 aircraft to flight assignments in a link is sufficient to model DARP. However, this assumption does not take into account that in short-haul operations there is a well-defined working day: At the end of the day, aircraft will lay over at various airports in order to carry out the planned early morning flights. It is therefore of prime importance that the correct number of aircraft lay over at each airport. In other words, a time horizon spanning the remaining part of the working day is necessary if such layover restrictions are present in the problem.

To cater for the layover restrictions, each link terminates in a sink node, thus indicating where the aircraft will lay over, e.g. aircraft 10 in Figure 1 ends its working day at airport 3. This makes it possible to keep track of the number of aircraft laying over at each particular airport.

Another feature in the underlying network is the cancellation aircraft node, e.g. nodes 3 and 7 in figure 1. By default, all cancellation aircraft nodes are assigned to the sink node. However, the cancellation aircraft can be assigned to any flight(s), thereby representing a cancellation of these. If for example cancellation aircraft 3 were swapped with aircraft 2, then aircraft 2 would remain at airport 1 and flights 2 and 6 would be cancelled.

Finally, there are surplus aircraft. By default, these are connected by a forward arc to the sink node. Consequently, if the surplus aircraft is not used, it remains at the airport. Alternately, it can be assigned to a flight, thus keeping it from getting cancelled or delayed.

4.2 Basic Parameters and Decision Variables

In the following, we formalize the components necessary to express the problem as a mathematical programming problem. These components are variables, parameters and index sets, and using these we are able to express the objective function used to quantify the cons of a solution, which as mentioned is necessary in any heuristics for the DARP-problem .

A = set of nodes representing aircraft.

a = index for aircraft nodes.

F = set of nodes representing flights.

f = index for flight nodes.

F_a = subset of F consisting of candidate flights considered for aircraft a. If aircraft a is delayed beyond the time horizon, F_a is set to empty. In Figure 1, F_a could reasonably consist of flights $\{4, 5, 6, 7\}$ for aircraft $a = 4$.

F_c = dynamic set containing flights f that have been cancelled.

r_f = the revenue of flight f.

d_{af} = the delay incurred if aircraft a is assigned to flight f. d_{af} is calculated as needed and takes all relevant previous assignments into consideration. d_{af} is measured in minutes.

α_f = delay cost multiplier associated with each flight f.

β_f = cancellation cost multiplier associated with cancelling flight f.

The decision variable is:

$$x_{af} = \begin{cases} 1 & \text{if aircraft } a \text{ is assigned to flight } f \\ 0 & \text{otherwise} \end{cases}$$

To interpretation of the decision variables, refer to Figure 1. Here $x_{1,1} = 1$ because aircraft 1 is assigned to flight 1. Conversely, $x_{1,8} = 0$ because aircraft 1 is not assigned to flight 8.

4.3 Objective function

Given the network and the basic parameters and decision variables, the objective function can be defined.

$$\begin{aligned} \mathit{Objective} = & \sum_{a \in A} \sum_{f \in F} r_f \cdot x_{af} \\ & - \sum_{a \in A} \sum_{f \in F \setminus F_c} \alpha_f \cdot DF \cdot r_f \cdot d_{af} \cdot x_{af} \\ & - \sum_{a \in A} \sum_{f \in F_c} \beta_f \cdot r_f \cdot x_{af} \end{aligned} \tag{1}$$

The first component in the objective function is the total revenue. The second component is the total cost of delays. The constant DF is the percentage of the revenue r_f which is subtracted per minute delay of flight f. The third component is the cost associated with cancellations.

In both the second and third component of the objective function, the revenue r_f is used directly to measure the cost because it seems a natural way to prioritize the flights. In principle this renders α and β unnecessary. However, a true revenue r_f is typically not calculated until several weeks after the flight has been flown – only then the necessary data are available. A revenue r_f calculated before the actual flight can only be based on forecasts, e.g. the number and types of passengers, the airports between which the flight is flown, the aircraft type carrying out the flight, etc. Disruptions to the flight schedule may render these forecasts obsolete, which is why α and β are relevant. These can be assigned values which guides a heuristic to find solutions that prioritize the flights according to the actual situation on hand.

4.4 Choice of Solution Neighborhood

Given the network representation of a schedule, a neighborhood can be defined. In short, the neighborhood to a DARP solution consists of all those solutions that can be reached by making 1 feasible swap between 2 different aircraft. By default, an

aircraft can be assigned to any flight departing from the airport where the aircraft is located. The size of this neighborhood is $O(n^2)$ where n is the number of flights. Notice that the neighborhood size can be halved by utilizing that only $\frac{n^2}{2}$ *different* neighbors exist.

4.5 Evaluating Solutions

The time it takes to explore a neighborhood depends heavily on how effectively the objective value is calculated for each solution. A reasonable way of doing this would be to track each aircraft through its link and calculate the contribution to the objective value. Cancellations would also be included this way, because cancellation aircraft are treated as a normal aircraft in this respect. In other words, each flight would be considered once resulting in a complexity of $O(n)$, where n is the number of flights. It is possible, however, to reduce this complexity significantly by only recalculating the cost of reassigned links.

Given the neighborhood size and the complexity of evaluating a single solution, the overall complexity of evaluating all solutions in a neighborhood is $O(n^3)$ where n is the number of flights. In practice, however, the observed complexity is significantly smaller.

4.6 Creating Problem Instances

To test the heuristics, which have been implemented to solve DARP, 25 problem instances were created using the generator outlined in Figure 3. Simply put, a link is created for each aircraft by repeatedly selecting a destination airport at random until the time horizon is exceeded. This limits the length of a link to a maximum of 5 flights. The resulting problem instances are listed in Table 1.

It is important to notice that because airports are randomly selected, flights will not be concentrated around a few airports like they would be in a hub-and-spoke system – instead they will be more evenly distributed. Likewise, aircraft will typically not travel back and forth between the same two airports. These 2 factors mean that the generated flight schedules do not resemble real flight schedules. However, it seems reasonable to assume that the complexity remains unchanged.

5 Choice of Heuristics

Several heuristics were implemented with the overall aim of finding that heuristic among these, which was able produce the best results in 3 minutes or less. Initially, various versions of a heuristic were implemented that had the possibility of escaping

```
procedure Test Instance Generator
    repeat
        Select an aircraft
        Select an initial ready-time
        Select airport randomly where aircraft will start
        Decide if the aircraft is delayed
        repeat
            Select a destination airport randomly
            Update time according to original ready-time
        until extending the link will exceed time horizon
    until the desired number of aircraft are in use
end
```

Figure 3. *Test instance generator outline.*

Instance No.	Number of Airports	Number of Aircraft	Number of Flights
1	10	20	80
2	10	30	125
3	10	40	166
4	10	50	206
5	20	20	78
6	20	40	158
7	20	60	231
8	20	80	306
9	20	100	382
10	30	30	111
11	30	60	221
12	30	90	326
13	30	120	440
14	30	150	559
15	40	40	148
16	40	80	290
17	40	120	432
18	40	160	588
19	40	200	741
20	50	50	195
21	50	90	333
22	50	110	404
23	50	150	562
24	50	200	753

Table 1. *Dimensions of the problem instances.*

local optima, namely the Iterated Local Search (ILS) with a Variable Neighborhood Search (VNS) incorporated. These methods are described in [Stü99] and [MH97] respectively and our use of them is thoroughly described in [LS01b]. A simple Steepest Ascent Local Search (SALS) heuristic was also implemented along with a repeated SALS (RSALS), which functions exactly like SALS but is repeated for different initial solutions. Surprisingly, the SALS algorithms were most effective.

5.1 Steepest Ascent Local Search Heuristic

The ILS algorithms were able to find reasonably good solutions quickly – typically within the first 10 seconds. Significantly better solutions were not found even in 24-hour test runs and with a wide range of different parameter settings. This prompts a number of questions concerning the search space structure and whether or not other types of heuristics would be more effective.

A SALS algorithm quickly finds a local optimum. However, once it has reached such an optimum, it is trapped. This is not a problem if the value of the local optimum is close to the value of the global optimum which turned out to be the case, cf. Section 6 and 7.

5.2 Implementing the SALS Heuristic

The simplified program structure of the SALS algorithm is shown in Figure 4. There are 3 main elements:

```
procedure Steepest Ascent Local Search (SALS)
    x_af = InitialSolution
    repeat
        x'_af = LocalSearch(x_af)
        x_af = AcceptanceCriterion(x_af, x'_af)
    until a better solution cannot be found
end SALS
```

Figure 4. *Outline of a SALS procedure*

InitialSolution: The initial solution is the original flight schedule including aircraft ready-times, scheduled departures, original tail assignments and all delays/cancellations. All these data are represented as illustrated in Figure 1.

LocalSearch: The local search procedure is initiated by a solution x_{af} in the form of a flight schedule. A best improvement strategy is chosen so that all of the neighbors to x_{af} are evaluated and the best solution x'_{af} among the neighbors to x_{af} is used as a starting point for the next iteration.

AcceptanceCriterion: This procedure determines if the latest local search iteration yields an improved solution. If so, it allows the algorithm to continue.

5.3 Solution Quality of SALS

In Table 2 the results achieved by the SALS algorithm are reported. For the sake of comparison, the column *Best Result* is introduced. This column refers to best solution ever found for each problem instance and it should be noted that these are not necessarily optimal. The best results have been found using the ILS-heuristic in 24-hour test runs with different parameter settings and in test runs where RSALS was repeated 2000 times using different initial solutions.

Instance No.	Number of Flights	Best Result	SALS	Gap (in %)	Time (in secs.)
1	80	264051	255213	3.35	0.09
2	125	402593	375699	6.68	0.12
3	166	614664	594911	3.21	0.50
4	206	690592	675224	2.23	0.90
5	78	201030	196291	2.36	0.03
6	158	508363	507043	0.26	0.26
7	231	814305	803711	1.30	0.63
8	306	1096463	1086028	0.95	1.29
9	382	1292384	1268061	1.88	3.16
10	111	376766	369218	2.00	0.02
11	221	782862	776675	0.79	0.34
12	326	1110793	1100121	0.96	1.12
13	440	1569761	1538055	2.02	2.62
14	559	1763103	1683127	4.54	6.77
15	148	437393	417642	4.52	0.07
16	290	963745	958332	0.56	0.57
17	432	1521328	1506960	0.94	1.70
18	588	1954775	1917987	1.88	5.19
19	741	2407012	2336639	2.92	12.15
20	195	612507	602827	1.58	0.15
21	333	1127078	1104572	2.00	0.63
22	404	1473042	1449719	1.58	0.85
23	562	1773247	1721653	2.91	4.05
24	753	2595359	2547788	1.83	9.42
Average gap between best solution and SALS:				2.22	(secs.) 2.11

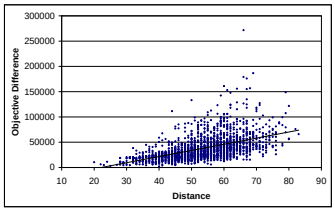
Table 2. *Overview of the SALS heuristic results*

SALS is clearly very fast. In Table 2, it takes 2.11 seconds on average to find a local optimum using SALS and never more than 13 seconds. It should be noted that the best solutions found to the problem instances listed in Table 1 improved the revenue in the original flight schedule by 59.0% on average given the parameters chosen. In other words, SALS improves the revenue approximately 57% on average.

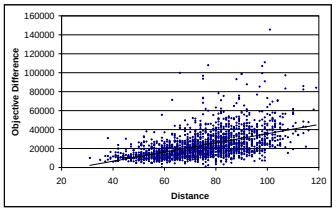
The ILS heuristics produced slightly better results than SALS in terms of revenue. However, the speed with which SALS finds solutions makes it very attractive. With respect to practical implementation at an airline, SALS would certainly be the most well-suited algorithm to solve DARP. This holds in particular because for ILS there is no apparent general correlation between the parameter settings and the quality of solutions found. This makes the tuning of ILS very difficult.

6 Analysis of the Search Space

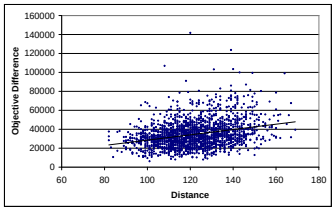
The speed with which SALS found a good solution calls for a further analysis of the nature of the solution space. It is unusual that a relatively simple SALS algorithm finds very good solutions compared to customized heuristics that run for 24 hours without getting trapped in local optima.



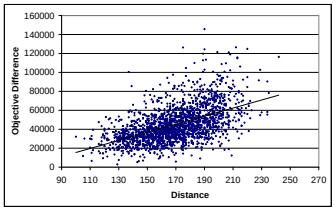
(a) Fitness landscape for problem instance 4



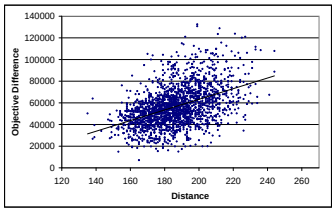
(b) Fitness landscape for problem instance 9



(c) Fitness landscape for problem instance 14



(d) Fitness landscape for problem instance 19



(e) Fitness landscape for problem instance 24

Figure 5. *Fitness landscapes for 5 of the problem instances listed in table 1.*

In Figure 5 fitness landscapes have been made for 5 problem instances (refer to Table 1). The data for these fitness landscapes were generated using RSALS mentioned

in section 5.

Each fitness landscape uses the best solution ever found for a particular problem instance as a reference point. Technically this reference point is the point (0,0) in each figure. All the other points represent other local optima found by RSALS. In all, there are 2000 such local optima in each fitness landscape. For each of these local optima the distance to the reference optimum is calculated. Here distance is a simple count of the number of aircraft to flight assignments in the local optima, which are different from the assignments in the reference optimum. This value is plotted against the numerical difference in the corresponding objective values. The result is a scatter diagram, which may show some correlation between the objective values and the distances.

In all the fitness landscapes in Figure 5 there is a remarkably clear correlation: The further the distance to the best solution ever found, the worse the objective values get. To illustrate this, a linear fit has been made in all the fitness landscapes. This fit does not explain so much of the observed variation. However, the tendency is clear and statistically significant. This correlation gives a strong indication that the solution space has a "hill-like" structure like that illustrated in Figure 6. Furthermore, the fact that SALS found such good revised flight schedules to all the problem instances also indicates that many local optima are close to global optimum, hence the wide "hill-top". If the solution space structure indeed looks like that of Figure 6, increased computational time is not going to improve the results significantly because better solutions do not exist once the "hill has been climbed". This kind of solution space structure is a strong argument for using SALS-like heuristics because the local optima, which SALS will be trapped in, have values close to that of the global optimum.

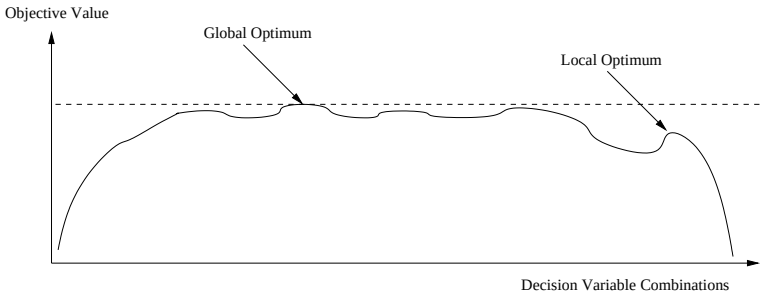


Figure 6. *Geometric fitness landscape as a function of all combinations of values assigned to the decision variables*

Notice that in each fitness landscape, the local optima seem to be grouped a certain minimum distance from the reference point. In Figure 5(a) there are no local optima with a distance of less than 20 to the reference optimum. A hypothesis explaining this observation is that the common distance between *all* local optima is at least 20 in Figure 5(a). This hypothesis is supported by the fact that the minimum distance to the reference optimum increases as the problem size increases: It seems reasonable

that the common distance between local optima in large problems is greater than that of small problems.

7 Testing the Heuristic on Real Problem Instances

As mentioned, the structure of the generated problem instances listed in Table 1 is not realistic. However, the results achieved using these instances were very promising. British Airways (BA) made some of their flight data available for realistic testing of the RSALS heuristic. These data were transformed to the network representation described previously. At this prototype phase the rules implemented are simplifications of the reality at British Airways.

More precisely, 10 BA flight schedules from November 2000 were extracted. Each flight schedule consists of all the BA short-haul flights that should have been flown a particular day including the associated tail assignments. On average, these test instances had the following characteristics:

- 80 active aircraft
- 44 airports
- 340 flights

Each flight schedule was disrupted by randomly delaying approximately 20% of the aircraft 30-240 minutes. The RSALS heuristic then attempts to improve the flight schedules by delaying flights, swapping aircraft to flight assignments and cancelling flights. The generated recovered flight schedule spans the remainder of the working day and respects the following:

- Aircraft balance
  - The number of aircraft originally intended to end in an airport must be respected.
- Swap at base only
  - Changes in tail assignments may only be made at the base airport (London Heathrow airport (LHR) in the case of BA).
- Respect minimum turnaround time
  - There is a minimum turnaround at the base airport (60 mins.) and for all outstations (30 mins.). Turnaround simply refers to minimum time required between landing an aircraft and taking off again.
- Only make feasible swaps with respect to aircraft types

  - Aircraft can only be swapped if they are of the same type, except Boeing 757 and 767, which can be swapped.

- Only cancel here-and-back trips
  - If cancellations are used, only pairs of trips are allowed to be cancelled, e.g. from LHR to Stockholm and back again.

7.1 Reference Test Results

To illustrate the effectiveness of RSALS and the influence of the parameters involved, the 10 problem instances described above have been improved using the following reference parameters:

$SwapCost$	5000
$CancellationCost$	2.0 · flight revenue (£)
$DelayCost$	0.025 ($\frac{1}{minutes}$) · flight revenue (£) · delay (minutes)

The Swap Cost does not represent the actual cost of swapping. It is a parameter used to control the number of swaps made by the DARP heuristic. The objective function does therefore in no way represent the actual costs of the revised flight schedule.

The average values of the test results from the reference test instances for all 10 days are shown in Table 3 using total delay in minutes and number of cancellations as key performance indicators. The delay is calculated over all flown flights, i.e. the cancelled flights do not contribute to this indicator. The delay is reduced by 62.1% on average and 2 cancellations are introduced. The heuristic modifies the lines of work of approximately 20 aircraft to recover the 16-17 lines of work that were delayed; approximately 10 aircraft end in an airport different from the one originally scheduled (end destination changes).

	Delay (in mins.)	No. of Canx.	No. of modified lines of work	End destination changes
Before	5961.7	0	0	0
After	2258.9	2	20.4	10.2
Difference	3702.8	2	20.4	10.2

Table 3. *Average values of the results over all 10 test instances.*

On average it took the RSALS heuristic 6.07 seconds to find the revised flight schedules. As mentioned, the RSALS heuristic is simply a repetition of the SALS heuristic for different initial solutions. In the case of the BA-flight schedules, SALS was repeated 5 times. Different initial solutions for the same flight schedule were created by introducing one random swap before initiating SALS.

7.2 Effects of Varying the Basic Parameters

In the following, the costs of the three basic options have been varied to illustrate the flexibility of the DARP-heuristic. The reference test instance is shown in boldface in all the following Tables.

7.2.1 Effect of Varying the Swap Cost

Swap cost	Delay (in mins.)	No. of canx.	No. of modified lines of work	End destination changes	CPU time (in secs.)
2000	2003.8	0.8	29.3	14.6	9.38
5000	**2258.9**	**1.6**	**20.4**	**10.2**	**6.07**
15000	2792.7	5.3	10.8	3.6	3.83

Table 4. *The effect of modifications to the swap costs.*

Varying the swap cost has a significant impact on the type of revised flight schedule generated by RSALS. In Table 4, a swap cost of 2000 (as opposed to 5000 in the reference test instances) increases the number of modified lines of work and end destination changes by 43.6% and 43.1% respectively. Lowering the swap cost corresponds to making changes in the flight schedule cheaper, thus the delay can be reduced by 66.9% as opposed to 62.1% in the reference test instances. In addition, a substantially smaller number of cancellations results.

As expected, increasing the swap cost has exactly the opposite effect. The number of modified lines of work and end destination changes is reduced by 47.1% and 64.7% respectively. The increased cost of swapping limits the number of attractive changes, thus the total delay is only reduced 53.2%. On average, 5.3 cancellations are introduced corresponding to the fact that compared with retimings, cancellations become more attractive.

Notice that the CPU time decreases with increased swap costs. This is to be expected: When swap costs are low, a larger number of changes in the flight plan are attractive, thus more CPU time is required before no swaps are found that yield an improved objective value.

7.2.2 Effect of Varying the Cancellation Cost

As mentioned in Section 4.3, the cost of cancelling a flight is a factor of the projected flight revenue. It is therefore expected that the number of cancellations made by the heuristic increases with a decreasing cancellation factor (see Table 5). Note, however, that the difference in delay minutes between solutions with many cancellations and solutions with no cancellations is modest. It is also demonstrated, that it is possible to effectively prohibit cancellations, if the cancellation factor is set to 10.0. The CPU time required does not vary with a varying cancellation factor.

Cancellation cost factor	Delay (in mins.)	No. of canx.	No. of modified lines of work	End destination changes	CPU time (in secs.)
1.0	1833.2	5.6	17.5	7.6	5.64
2.0	**2258.9**	**1.6**	**20.4**	**10.2**	**6.07**
5.0	2577.6	0.4	20.8	10.6	5.97
10.0	2679.9	0.0	21.1	9.8	6.68

Table 5. *The effect of modifications to the cancellation cost factor.*

7.2.3 Effect of Varying the Delay Cost

Delay cost factor	Delay (in mins.)	No. of canx.	No. of modified lines of work	End destination changes	CPU time (in secs.)
0.01	3676.8	0	13.2	5.3	3.64
0.025	**2258.9**	**1.6**	**20.4**	**10.2**	**6.07**
0.05	1540.7	6.4	21.0	10.8	6.95

Table 6. *The effect of modifications to the delay cost factor.*

The delay cost is a percentage of a flights revenue for every minute it is delayed. Lowering this percentage corresponds to making delays more acceptable, which is clearly seen in Table 6: With a delay cost factor of 1%, the total delay is only reduced by 38.2% as opposed to 62.1% in the reference test instances. Conversely, a delay cost factor of 5% reduces the delay by 74.2%. However, a large delay cost factor also makes cancellations relatively more attractive, which is why 6.4 cancellations are introduced in this case, thus again clearly illustrating the trade-off between the two options.

7.2.4 Search Space Analysis of the BA Test Instances

In section 6, fitness landscapes were constructed, which indicate that the search space has a hill-like structure that enables a SALS-heuristic to find good solutions. To investigate if the BA test instances have the same type of structure, fitness landscapes have been generated using the same method as before. The conclusion is that in the BA test instances, the further the distance to the best solution ever found, the worse the objective values get (see [LS01a]). Furthermore, the average gap between the best solutions and the reference test instance solutions is 2.99%. It is therefore clear that in terms of using SALS-heuristics, the theoretical test instances and the BA test instances are very similar. The best solutions ever found to each of the BA test instances were again found using RSALS repeated 2000 times.

8 Conclusion

This article demonstrates that the Dedicated Aircraft Recovery Problem (DARP) can be solved effectively by a relatively simple SALS-heuristic using the underlying network representation of the problem. On average, less than 10 seconds are required to find a feasible revised flight schedule that includes all planned flights on a given day. Furthermore, our method allows the necessary flexibility that flight controllers need in terms of creating different types of revised flight schedules. Particularly, this is demonstrated using real test instances from British Airways. In 10 such instances, our method of solving DARP effectively decreased the total delay by introducing swaps and cancellations. The effect of applying various strategies was also shown: Depending on the situation at hand, the trade-off between delays, cancellations and swaps can be adjusted and our SALS-heuristic can generate feasible revised flight schedules that accommodate these trade-offs in less than 10 seconds on average.

Our work gives a strong indication that DARP is a relatively simple problem to solve when represented as described. Hence, the future challenge lies in incorporating crew and passenger considerations into a tool for integrated recovery of aircraft, crew, and passengers, which is the ultimate goal of the Descartes project. The aircraft recovery module of the current Descartes software is based in the ideas and observations of the current paper.

9 Acknowledgements

We would like to thank Alex Ross and Nicki Davis from British Airways who supplied us with airline industry insights and evaluated and commented our results. Likewise, we would like to thank Allan Larsen from Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark for commenting and assisting.

This work is partly supported by the EU-funded project DESCARTES.

References

[ABY97] M. F. Argüello, J. F. Bard, and G. Yu. A grasp for aircraft routing in response to groundings and delays. *JCO*, 5:211–228, 1997.

[ABY98] M. F. Argüello, J. F. Bard, and G. Yu. Models and Methods for Managing Airline Irregular Operations. In Gang Yu, editor, *Operations Research in the Airline Industry*. Kluwer Academic Publishers, Boston, 1998.

[CHLL01] Jens Clausen, Jesper Hansen, Jesper Larsen, and Allan Larsen. Disruption management – operations research between planning and execution. *OR/MS Today*, 28(5):40 – 43, October 2001.

[CK97] J.-M. Cao and A. Kanafani. Real-Time Decision Support for Integration of Airline Flight Cancellations and Delays Part I & II. *Transportation Planning and Technology*, 20:183–217, 1997.

[DLT01] Nicki Davis, Allan Larsen, and Sergey Tiourine. Descartes – decision support for integrated crew and aircraft recovery. Presentation at the AGIFORS Airline Operations Study Group Meeting in Jamaica, May 2001.

[JYKR93] A. I. Z. Jarrah, G. Yu, N. Krishnamurthy, and A. Rakshit. A Decision Support Framwork for Airline Flight Cancellations and Delays. *Transportation Science*, 27:266–280, 1993.

[LS01a] M. Løve and K. R. Sørensen. Dedicated aircraft recovery using heuristic methods - tested on british airways flight schedules. Technical report, Informatics and Mathematical Modelling (IMM). Techical University of Denmark (DTU), May 2001. Product of the DECARTES project.

[LS01b] M. Løve and K. R. Sørensen. Disruption management in the airline industry. Master's thesis, Informatics and Mathematical Modelling (IMM). Techical University of Denmark (DTU), March 2001. http://www.imm.dtu.dk/documents/ftp/ep2001/ep16_01-a.html.

[MH97] N. Mladenović and P. Hansen. Variable Neighborhood Search. *Computers & Operations Research*, 24:1097–1100, 1997.

[Stü99] T. Stützle. Iterated Local Search for the Quadratic Assignment Problem. Technical report, Darmstadt Technische Hochschule, 1999.

[TG84] D. Teodorović and S. Guberinić. Airline Optimization. *European Journal of Operational Research*, 15:178–182, 1984.

[TS90] D. Teodorović and M. Stojković. Model for Operational Daily Airline Scheduling. *Transportation Planning and Technology*, 14:273–285, 1990.

[YY96] S. Yan and D.-H. Yang. A Decision Support Framework for Handling Schedule Pertubations. *Transportation Research*, 30:405–419, 1996.