

Air Traffic Flow Management with Heuristic Repair

Ulrich Junker

*ILOG, 1681, route des Dolines, F-06560 Valbonne,
E-mail: uli.junker@free.fr*

Abstract

The European air traffic flow management problem poses particular challenges on optimization technology as it requires detailed modelling and rapid online optimization capabilities. Constraint programming proved successful in addressing these challenges for departure time slot allocation by offering fine-grained modelling of resource constraints and fast allocation through heuristic-repair strategies.

1 Introduction

Air traffic congestion remains an important problem in European and North-American airspaces and is usually solved by delaying the departure of flights. A slot allocation or ground-holding policy assigns a departure time to each flight such that congestion is avoided while ensuring other important objectives such as safety, equity, and the reduction of the total delay. Whereas it is straightforward to find some slot allocation policy that avoids congestion by iteratively delaying flights that traverse overloaded sectors, it is not evident to understand whether such a policy well meets the other objectives as well. A first difficulty consists in expressing general objectives such as safety and fairness in terms of clearly defined properties of the slot allocation policies. A second difficulty consists in understanding the interactions between the objectives and in defining trade-off preferences between different slot allocation policies. For example, should a policy that has a smaller total delay be preferred to a policy that is more equitable? A third difficulty consists in understanding whether a given slot allocation policy is efficient or whether there are significantly better policies given the preferences defined before.

In order to study certain of those questions, the Eurocontrol Experimental Centre has developed a system for innovative slot allocation, cf. Dalichampt *et al.* (1997) by using constraint programming (see Rossi *et al.* (2006)) and integer programming methods (see Wolsey & Nemhauser (1999)). The application of optimization technology to the slot allocation problem provided several challenges. Firstly, it was necessary to develop an appropriate model of the slot allocation problem. Existing approaches, cf. Bertsimas & Stock (1994), were adapted to the North-American ATFM problem and did not take the specific safety and equity notions of the European problem into account. Secondly, it was necessary to develop fast allocation algorithms that provided nearly optimal results for problem instances of large size (10000 flights) in a short time. Constraint programming has proved successful in addressing these challenges by providing fine-grained modelling of resource constraints and fast allocation through heuristic-repair strategies, cf. Minton *et al.* (1992). Whereas the study in Dalichampt *et al.* (1997) introduces the constraint-based slot allocation problem and discusses the results of the constraint-based slot allocation approach from an ATFM perspective, this paper explains the problem-solving principles that helped to achieve the results reported in Dalichampt *et al.* (1997). The present work thus complements the study in Dalichampt *et al.* (1997) and gives insights into the problem solving

methods that successfully solved a particular definition of the ATFM problem without claiming that these results carry over to current and future evolutions of the European ATFM problems.

Other applications of constraint programming to ATFM problems add flexibility in the workload model of the regulated sectors (see Barnier *et al.* (2001)) and address the problem of resolving traffic complexity within and around a sector (see Flener *et al.* (2007)).

The paper introduces the constraint-based slot allocation problem in section ?? . Section ?? describes approximate and exact formulations of this problem within constraint programming and integer programming. Section ?? describes the heuristic repair strategy used by the constraint programming approach and section ?? evaluates the methods by a real-world ATFM example.

2 A Capacity-oriented Slot Allocation Problem

We briefly describe the slot allocation problem as it has been defined by the Eurocontrol Experimental Centre in 1997, cf. Dalichampt *et al.* (1997). This problem avoids congestion of airspace sectors by delaying the take-off of flights while keeping the flight route and flight speed as planned. The planning horizon of this slot allocation problem is one day of traffic. It is solved offline before each day of operation as well as online during the day of operation to take into account changes in flight plans and sector capacity.

2.1 En-route Capacity Constraints

The airspace is divided into a set of m sectors $\mathcal{S} := \{s_1, \dots, s_m\}$. These sectors are traversed by n flights $\mathcal{F} := \{f_1, \dots, f_n\}$ for a given day of traffic. Each flight f_i has an expected take-off time and a travel plan which is a sequence of sectors $s_{i,1}, \dots, s_{i,m_i}$. The travel plan specifies when the flight f_i is expected to enter the sector $s_{i,j}$ in terms of the expected time over $eto_{i,j}$. This travel plan is strict and does not allow delays or speed-ups when the flight plan is airborne.

Certain sectors are subject to regulations that limit the number of flights that can enter these sectors during each hour within a given time period. This time period is divided into successive intervals of one hour length and thus several capacity constraints are obtained for each regulated sector. The k -th capacity constraint for sector s_j is specified by a capacity $c_{j,k}$ and a time period $[s_{j,k}, e_{j,k})$. The constraint is satisfied if the number of flights entering the sector s_j during the interval $[s_{j,k}, e_{j,k})$ is smaller than or equal to $c_{j,k}$. The set F_j denotes the set of flights that enter the sector s_j :

$$\| \{i \in F_j \mid s_{j,k} \leq d_i + eto_{i,j} < e_{j,k}\} \| \leq c_{j,k} \quad (1)$$

A slot allocation policy assigns a non-negative delay d_i to each flight f_i such that all capacity constraints are satisfied. Negative delays due to departures ahead of schedule are not allowed. The total delay of a slot allocation policy is the sum of the d_i 's for all flights f_i . Policies with smaller total delay are preferred.

2.2 Discussion of the First-Come First-Served Principle

Other objectives of Eurocontrol's slot allocation methods are equity and safety. In 1995, a first-come first-served algorithm has been used to allocate slots to the flights that enter a sector by Eurocontrol's operational ATFM system. This principle preserves the order in which the flights enter a sector by allocating earlier flights first. This principle can be respected for all sectors and flights if each flight traverses only one regulated sector. In realistic scenarios, the flights are traversing multiple sectors and those sectors may impose different orders among the same flights. For example, a flight f may precede another flight f' in a regulated period of a sector s and f' may precede f in a regulated period of another sector s' . If only one flight can enter those sectors during the regulated period, then one of both flights has to be delayed. If the first-come first-served principle is applied to sector s , then the flight f' may be delayed. Delaying f' in sector s will also delay it in sector s' . If the delay is sufficiently large, f' will no longer precede flight f in sector s' , meaning that the first-come first-served principle is violated for sector s' .

As a consequence of this interaction, a complete application of the first-come first-served equity principle is not possible. The practice of 1995 was to apply it to the most penalizing regulation only.

One of the purposes of the study in Dalichampt *et al.* (1997) was to measure the cost of this relaxed equity principle in terms of the total delay. Eurocontrol wanted to know how much the total delay can be reduced if this equity principle is not applied. For this purpose, the first-come first-served principle has not been included in the constraint model. Although equity is a more important objective than the minimization of the total delay, it is necessary to first define when one slot allocation policy is more equitable than another one before incorporating this objective into a combinatorial optimization approach. Equity principles such as those studied in collaborative decision making and computational social choice theory (see Yann Chevaleyre *et al.* (2007) for an introduction) may provide help for establishing such a definition, but are beyond the scope of this paper.

2.3 Smoothing Constraints for Safety Considerations

A reduction of the total flight delay will only be acceptable if a sufficient level of safety is guaranteed. Even if a sector has a capacity of 30 flights per hour, it will not be acceptable if all 30 flights enter the sector at the same time as this will lead to peaks in the workload of the sector. For safety reasons, it is desirable to "smoothen" this workload and to allow a single flight each two minutes. The need for smoothing arises since the contractual capacity constraints limit the number of flights that enter a sector during each hour, but do not limit the number of flights that traverse a sector simultaneously as it is, for example done in alternative approaches such as Bertsimas & Stock (1994).

The practice of Eurocontrol consisted in dividing a regulated period of a sector into time slots of equal length such that the number of slots corresponds to the capacity of the period. A single flight can be allocated to each slot. If a flight is allocated to a slot, then it should enter the sector during the duration of this slot. In other words, slots are capacity constraints of capacity 1. These capacity constraints imply the contractual capacity constraints of the duration of one hour.

If each flight traverses a single regulated sector, this sector will determine the delay of the flight and allocate it to a slot. In this case, the slot capacity constraint can well be respected. However, if a flight traverses several regulated sectors of different capacities, then the durations of the slots of these sectors differ. If a flight is allocated to a slot in sector s , then it may enter a sector s' during a slot which is already occupied even if the preceding slot is available. To respect the slot capacity, it is then necessary to delay the flight again. This procedure may lead to high delays due to rigorous satisfaction of the auxiliary slot constraints. Hence, dividing the hourly capacity into time slots of capacity one achieves complete smoothing, but leads to an additional delay. The operational system (see the CFMU (2009)) applies a more relaxed allocation policy which was difficult to model.

However, smoothing can also be achieved by dividing the hourly capacity by a fixed factor k and by setting up k sub-periods of equal length. For example, the 30 flights per hour can be distributed over 12 sub-periods of the length of 5 minutes. Each of these sub-periods has a capacity of 3 flights, meaning that 3 flights can enter the sector during a sub-period. It is important to understand that these smoothing constraints do not imply the hourly capacity constraints as the capacities of the sub-periods are systematically rounded up. In the example above, each of the 5-minute sub-periods has a capacity of 3, meaning that the 12 sub-periods together lead to a total capacity of 36 instead of the contractual capacity of 30 flights per hour. Therefore the smoothing constraints have to be used in addition to the hourly capacity constraints, but do not replace them.

This technique allows a flexible way of smoothing and permitted Eurocontrol to study the impact of smoothing on the total delay, cf. Dalichampt *et al.* (1997), which is in particular interesting since there is no clearly defined numerical measure for safety.

2.4 Online optimization

The slot allocation problem as described above is solved offline before the day of operation. The resulting slot allocation policy will be subject to modifications due to unexpected events during the day of operation such as canceled flights or modification of sector capacity. For these reasons, an online version of the slot allocation problem will be solved in regular intervals during the day of operation. This online version does not change the delay of those flights that are planned to take off within a fixed time period from the planning time on, but it may choose a completely new delay for all the other flights. The online optimization thus revises the future part of the slot allocation policy, while preferring revised policies with a smaller total delay. The online optimization thus limits the impact of unforeseen events on the total delay.

3 Constraint Models for the Slot Allocation Problem

Whereas the previous section has defined the slot allocation problem in terms of constraints and objectives, this section formulates this definition within integer and constraint programming, which are both general paradigms for solving combinatorial problems. The problem formulation requires the choice of a time scale for the flight delay, the expected time-over of a flight in a sector, and the start and end of each sub-period that is subject to capacity constraints. This time-scale thus transforms the delay d_i into a rounded delay d'_i , its expected time-over $eto_{i,j}$ into a rounded expected time-over $eto'_{i,j}$, and the start and end times $s_{j,k}, e_{j,k}$ of a sub-period into rounded start and end times $s'_{j,k}, e'_{j,k}$. We say that a formulation of the ATFM problem is exact iff it preserves the constraints as defined. This means that the condition $s'_{j,k} \leq d'_i + eto'_{i,j} < e'_{j,k}$ is true whenever the original condition $s_{j,k} \leq d_i + eto_{i,j} < e_{j,k}$ is true. Whereas constraint programming allows an exact formulation without increasing the size of the constraint model, this is not true for integer programming, where this exact formulation has an additional cost.

3.1 Exact Formulation within Constraint Programming

The slot allocation problem as described so far can be modelled with constraint programming. For each flight, we will introduce an integer variable that represents the flight delay. Its lower bound is 0 and its upper bound is set to a maximal delay that can be attributed to a flight. These delay variables are subject to capacity constraints of the form (??). We introduce capacity constraints for the contractual capacity constraints fixing the hourly capacity of a regulated period as well as the additional smoothing constraints. Furthermore, we introduce a variable for the total delay and a constraint that fixes it to the sum of the flight delays. Constraint programming allows a very precise modelling of the problem since a sufficiently small time scale can be chosen without increasing the size of the model. Indeed, we can choose a time-scale that is sufficiently small to guarantee an exact formulation. As said above, we need to be sure that the chosen time-scale preserves the conditions $s_{j,k} \leq d_i + eto_{i,j} < e_{j,k}$ used in the capacity constraints. Whereas the expected time-over $eto_{i,j}$ is expressed in minutes, the start and end times $s_{j,k}, e_{j,k}$ may require a finer time granularity if sub-periods of 2 and a half minutes are chosen. In order to permit this flexibility, we choose a time granularity of one second, although a time granularity of 1 minute is sufficient in most cases.

Constraint programming also permits a modeling of negative flight delays by enlarging the domain of the delay variables and of speed-ups or delays when the flight is airborne by introducing variables and constraints for the expected-time-overs of the flights. However, as the flight plans in the study of Dalichampt *et al.* (1997) are strict, this richer constraint model and its impact on the objective function and the solving method have not been investigated in the scope of this work.

3.2 Approximate and Exact Formulation within Integer Programming

The slot allocation problem can also be formulated as an integer programming problem by following the approach in Bertsimas & Stock (1994) and Bertsimas *et al.* (2008). In order to formulate the capacity constraints, it is necessary to replace the delay variables d_i by binary variables that permit to determine whether a flight f_i enters a sector s_j during period k which is the case iff the condition $s_{j,k} \leq d_i + eto_{i,j} < e_{j,k}$ is true. Bertsimas' technique consists in discretizing the time and to choose a time step Δ , e.g. of 5 minutes. We then introduce a binary variable $d_{i,t}$ for each time t that is a multiple of Δ and that is smaller than the maximal delay. This variable has the value 1 iff the flight f_i has at least the delay t . The binary variables $d_{i,0}, d_{i,\Delta}, d_{i,2\Delta}, \dots$ represent the delay of the flight iff the constraints $d_{i,t} \geq d_{i,t+\Delta}$ are satisfied for all relevant times t . The capacity constraints can then be formulated as follows:

$$\sum_{i \in F_j} (d_{i,r_\Delta(s_{j,k}-eto_{j,k})} - d_{i,r_\Delta(e_{j,k}-eto_{j,k})}) \leq c_{j,k} \quad (2)$$

The function r_Δ truncates its argument to multiples of Δ . The objective of minimizing the total delay is achieved by minimizing the linear expression $\sum_{i=1}^n \sum_t \Delta \cdot d_{i,t}$. This integer programming is no longer exact as the time granularity of 5 minutes does not guarantee the preservation of the conditions of the form $s_{j,k} \leq d_i + eto_{i,j} < e_{j,k}$. A further drawback of this integer programming model is that its size depend on the chosen time scale. A smaller time step Δ will provide more precise results, but leads to a larger number of variables. For a maximal delay of two hours and a time step of 5 minutes, the model needs 24 binary variables for each flight. Hence, 240000 binary variables are needed for problems involving 10000 flights without guaranteeing precise results.

An exact integer programming model can be obtained by removing the rounding operations in the constraint (??) and by introducing the binary variables $d_{i,s_{j,k}-eto_{j,k}}$ and $d_{i,e_{j,k}-eto_{j,k}}$ which are needed without the rounding. In general, this will significantly increase the number of variables needed per flight.

Constraint programming provides exact modelling of the slot allocation problem as defined before while keeping the size of the model proportional to the problem description. Integer programming leads to significantly larger models even if a less precise time granularity is chosen. Hence, constraint programming combines fine-grained modelling with a compact representation of this model and was chosen as the core technology for this study. Integer programming was used to evaluate the optimal total delay for smaller scenarios and permitted to assess the quality of the constraint programming solving strategies, which will be presented in the next section.

4 Problem Solving with Heuristic Repair

The nature and size of Eurocontrol's slot allocation problem required to carefully design an adequate solving strategy. Realistic instances involve twenty-thousand flights of which 20% to 25% may be regulated. Furthermore, changes during the day of operations requires frequent replanning within intervals of five minutes or less. Hence, the available running time was five minutes. An important purpose of the study was to show whether optimization methods are able to produce results within this time frame that are better than those of the operational system, which allocated flights based on the first-come first-served principle.

4.1 Minton's Heuristic Repair Method

As the problem is combinatorial in nature, adequate solving techniques had to be designed. As the flight plans are fixed, the problem has a particular mathematical structure and cannot easily be decomposed into subproblems. This distinguishes this particular ATFM problem from other problems which allow delays and speed-ups of a flight when it is airborne. As this is not allowed, delaying a flight necessarily impacts all the sectors that are traversed by the flight. Hence, it is not possible to reduce overloaded sectors during a fixed time period without modifying the workload

of other sectors outside of this chosen time period. Hence, it was necessary to design a heuristics that solves the problem as a whole by a sequence of globally good decisions. The heuristic-repair principle, cf. Minton *et al.* (1992), provides an adequate framework for elaborating such a method. The heuristic-repair method uses a current assignment of values to variables. It determines which constraints are satisfied under this assignment and which are violated. It is possible to associate a degree of violation to each constraint with the idea that higher degrees of violations are more difficult or more costly to repair. The difficulty of the whole problem is obtained by summing up the degrees of violations for each constraint.

In each iteration, the heuristic-repair method chooses a violated constraint and tries to reduce its degree of violation. For this purpose, the method inspects the variables of this constraint. It seeks to change the value of one of those variables by choosing another value from the domain of this variable. It evaluates the impact of this change on the problem as a whole and it chooses the assignment which achieves the highest reduction of the global difficulty. It then changes the assignment correspondingly. Then this step is iterated until no violated constraint remains anymore. Hence, heuristic repair, in its basic form, is essentially a local search method. Its drawback is that it is not guaranteed to terminate since it may run into cycles.

However, Minton has also shown that this heuristic repair can be achieved in the scope of a Branch-and-Bound method as it is common in constraint programming solvers. This integration can be achieved by maintaining a current value and a current domain for each variable. The current value must belong to the current domain. When changing the value of a variable x from v to v' , the method is branching. In the left branch, the decision $x = v'$ is applied. As a consequence, v' becomes the current value of x and the degree of violations of all constraints are updated. In the right branch, the decision $x = v'$ is rejected, meaning that the value v' is removed from the domain of x . As a consequence, it cannot be chosen any more in the right subtree. The value v is kept as current value in the right branch.

4.2 Least-Committing Heuristic Repair

As observed by Minton, this method can quickly and frequently run into dead-ends since a repair decision made in one step may be questioned in subsequent steps. In the case of slot allocation, the repair action consists in fixing the delay of a flight that crosses an overloaded sector. Once the delay of a flight is fixed, the flight cannot be delayed any more in further repair actions occurring in the left subtree even if this appears to be desirable. This problem can be avoided by a different branching policy. Instead of assigning the new value v' to the variable x , we remove its current value v . In the left branch, the decision $x \neq v$ is applied. As a consequence, the value v is removed from the domain of x and cannot be chosen any more in the left subtree. Furthermore, the new value v' is chosen as the current value in the left subtree. In the right subtree, the decision $x \neq v$ is rejected meaning that the value v is assigned to x in the right branch. As such the method automatically keeps the current value in the right subtree. As a consequence of this new policy, the current value of a variable can be changed several times on a left descent, thus making the behaviour of a left descent closer to the original local search version of heuristic repair while guaranteeing termination.

Compared to Minton's original strategy, this new least-commitment strategy thus inverts the form of the left and right branches. Instead of assigning a value in the left branch, it removes a value from the domain. As the left branch is explored first by a chronological search strategy, this change has a strong impact on the order in which solutions are found. The second, more important difference consists in the fact that the least-commitment strategy does not use the new value v' of the variable x when branching on x , but its current value v . It is important to understand that the current values are used to evaluate the constraint violations and thus determine the choice of the repair action. As such, the values v and v' have different roles. Hence, not only the least-commitment strategy inverts the left and right branches, it also manages the current values differently. The least-commitment strategy repairs a constraint violation by eliminating one of

the current values that are causing the violation or it assigns this value when the repair action fails. In contrast to this, Minton's original strategy repairs a constraint violation by assigning a new value which replaces one of the current values causing the violation or it eliminates this new value when the repair action fails. The difference is subtle, but it nevertheless has a strong impact on the behaviour of the whole method.

We apply this modified heuristic-repair strategy to the slot allocation problem. We use the lower bound $lb(d_i)$ of a variable d_i as its current value. This reflects the fact that the lower bound is the preferred value for the delay variable. Based on this current delay, we define the current load of a regulated period as follows:

$$L_{j,k} := \|\{i \in F_j \mid s_{j,k} \leq lb(d_i) + eto_{i,j} < e_{j,k}\}\| \quad (3)$$

An overload is obtained if this load exceed the capacity:

$$O_{j,k} := \max(L_{j,k} - c_{j,k}, 0) \quad (4)$$

The difficulty of the problem is then defined as the total overload, i.e. the sum of the overloads of all constraints.

The heuristic-repair method for the slot allocation problem then selects an overloaded capacity constraint in each step. It determines the flights that contribute to this overload and seeks to reduce it by increasing the delay of one of these flights. If f_i is one of those candidate flights, $e_{j,k}$ is the end time of the capacity constraint, and v_i is the current delay of f_i , then the flight delay has to be increased from v_i to $e_{j,k} - eto_{i,j}$ in order to ensure that f_i does no longer enter the sector s_j before time $e_{j,k}$. The method determines this new delay for each of the candidate flights. It then evaluates the impact of each of these repair actions on the total overload. It selects the repair action that achieves a good compromise between a high reduction of the total overload and a low increase of the delay. As described above, branching is done before applying the repair action. In the left branch, the decision $d_i \geq e_{j,k} - eto_{i,j}$ is applied. As a consequence, all delays smaller than $e_{j,k} - eto_{i,j}$ are removed from the current domain of the variable d_i and the value $e_{j,k} - eto_{i,j}$ automatically becomes its new current value. In the right branch, all delays that are greater than or equal to $e_{j,k} - eto_{i,j}$ are removed from the current domain of variable d_i and its current value remains unchanged. In the right branch, flight f_i can no longer be used in repair actions for reducing the overload of the selected capacity constraint. If the number of flights that will surely enter the sector during the regulated time period is equal to its capacity, then constraint propagation will reduce the domains of the remaining flights and ensure that they cannot enter the sector during this time period.

Hence, this method successively reduces overloads by increasing the delay of flights in a left descent of the search tree. Usually, the first solution obtained in this way is already of a good quality, meaning that no backtracking steps occur in practice to find the first solution if no maximal delay is imposed. If there is a maximal delay, it will impose an upper bound on the delay variables, meaning that repair action are failing if they assign a new delay that exceeds this maximal delay. As such, a maximal delay may require backtracking steps to find a first solution. Usually the first solution has such a good quality that a significant part of the search tree need to be traversed in order to find a better solution by chronological backtracking. Limited discrepancy search (see Harvey & Ginsberg (1995)) should behave better in this case, but was not tried out during the study.

Another conceptually simpler method is chronological scheduling for the CP model of subsection ???. This method visits the flights in increasing order of their expected take-off times and assigns the smallest possible delay in each step. Constraint propagation may reduce the domains of other variables as a consequence of these assignments. In general, the heuristic-repair method produces better results than the chronological method.

5 Evaluation of the Modeling and Solving Methods

Whereas Eurocontrol conducted ATFM studies based on the constraint-based slot allocation approach, this section provides a first evaluation of the different modeling and solving methods. The experiments are conducted for an example reported by Eurocontrol in 1995. It corresponds to one day of traffic over France and involves 16 regulated sectors and about 2000 flights that traverse those regulated sectors. As not all sectors have regulations, these numbers include only a percentage of the flights and sectors. Although this example is smaller than the whole European ATFM problem, it permitted to establish a first experimental comparison of the different methods and an understanding of their characteristics. It seeks to draw conclusions about problem formulation and solving, but not about air traffic flow management. We therefore refer the reader to the study in Dalichampt *et al.* (1997) for the ATFM results obtained for the larger problems.

A first experiment compared the different modelling approaches, namely the approximate and exact IP model and the exact CP model with respect to size, i.e. the number of variables and constraints. A maximal delay of 2h has been introduced to limit the size of the IP models. This experiment used a smoothing of the hourly capacity over sub-periods of 10 minutes, but did not include the hourly capacity constraints themselves. The approximate IP model used a time scale of 10 minutes and had 22925 binary variables and 21604 constraints. The exact IP model increased these numbers to 32079 binary variables and 30755 constraints. The exact CP model consisted of 1989 delay variables and 16 global constraints regrouping the capacity constraints of each of the 16 sectors. These numbers show that the usage of integer programming becomes prohibitive for larger problems as the number of variables is proportional to the number of flights. Interestingly the heuristic repair method in form of the overload reduction method found a total delay which is only 15% away from the optimum of the linear programming relaxation of the exact IP model. Astonishingly this total delay is 10% better than the optimum of the linear relaxation of the approximate IP model. Hence, the results of the heuristic repair method appeared to be better than an IP approach that followed standard modelling practices. This observation has been made at a late state of the project, but is certainly worth to be explored in more detail.

A second experiment compared the different solving methods and smoothing constraints of the CP model in terms of the total delay. The experiment imposes the contractual hourly capacity constraints as well as smoothing constraints on intervals of 10 minutes. Again the delay of each flight is limited to 2 hours. The chronological method finds a solution of a total delay of 32401 minutes within 0.17 seconds on a Dell Precision 340. The overload reduction method reduces this delay by about 30% to 21267 minutes (found in 0.69 seconds). If only the contractual capacity constraints are used and the smoothing constraints are switched off, then the total delay found by the chronological strategy drops to 19441 minutes (found in 0.15 seconds). The overload reduction method finds a total delay of 12492 minutes in this case (while needing 0.26 seconds). Finally if only the smoothing constraints are used and the contractual constraints are switched off, then the chronological strategy finds a delay of 16887 minutes and the overload reduction method finds a delay of 11537 minutes (in 0.5 seconds). These experiments show that the heuristic repair method significantly reduces the total delay compared to a standard chronological strategy. They also show the high impact of the smoothing on the result, meaning that it has to be chosen with care. The effects of the smoothing on the workload profile has not been studied.

This very first evaluation showed that the heuristic repair method for the exact CP model with smoothing constraints is a viable approach to slot allocation. This method was therefore used to carry out the study in Dalichampt *et al.* (1997). This study observes that the relaxation of the first-come first-served principle provides an interesting alternative to slot allocation as it reduces the total delay compared to the CASA algorithm while keeping a similar smoothing.

6 Conclusion

Eurocontrol has studied innovative slot allocation algorithms based on constraint programming in the years 1995 to 1997 and the results of the study can be found in Dalichampt *et al.* (1997). In this paper, we have described the principles behind these algorithms and have explained why constraint technology has been able to produce slot allocations of a smaller total delay than those found by the first-come first-served method and this within a sufficiently short running time to perform online allocation during the day of operation.

On the one hand, constraint programming offers detailed modelling capabilities which permitted an exact representation of the flight times and sector regulations. As all times are exactly represented, no extra delay is introduced by rounding operations that adapt the times to a scale of 5 minutes or so. This absence of rounding constitutes a significant advantages over integer programming models, cf. Bertsimas & Stock(1994), which need rounding in order to keep the size of the models reasonable.

On the other hand, constraint programming also allows a fast construction of a first solution of good quality by using a tailored heuristic-repair method. This method uses a current solution in addition to the current domains and is driven by the goal of reducing current sector overloads. It achieves this goal by delaying one of the flights that contributes to one of the overloads. As flights can cause overloads in several sectors, it prefers to delay those flights that provide a best total reduction of the overloads. Hence, the method exploits the relationship between overloaded sectors and flight delay to make decisions that reduce the difficulty of the remaining problem as much as possible. The method can be adapted to other additive optimization objectives such as minimizing the delay per passenger by introducing appropriate weights for the flight delays.

Although heuristic repair basically is a local search method, it has been embedded in a systematic backtrack search such that several delay actions can be applied to the same flight in a left descent of the search tree. The method is also able to backtrack if it encounters a dead-end during the left descent which may happen since a maximal flight delay is imposed. The method could further be improved by a final improvement phase as certain delays may be unjustified in the end. It may indeed happen that a flight is delayed in order to reduce an overload. However some other flight contributing to this overload may be delayed by some other step of the heuristic repair method, meaning that the delay of the first flight is no longer necessary. A final improvement phase can detect these unjustified delays and reduce them, thus ensuring that each flight delay is justified in the end.

This paper has thus provided further insights into the constraint-based slot allocation approach used in the study of Dalichampt *et al.* (1997). It explains why this approach successfully solved a particular definition of the ATFM problem. The work provided interesting lessons for solving combinatorial problems and showed that a least-commitment heuristic repair strategy is very effective if a problem can be solved by successively increasing lower bounds of variables. Finally it showed that constraint programming is a viable approach to ATFM problems and thus enriches an emerging literature on CP-based approaches to ATFM problems, cf. e.g. articles of Barnier *et al.* (2001) and Flener *et al.* (2007).

The work also showed that important objectives such as equity cannot easily be modeled in the constraint-based slot allocation approach and require a deeper investigation. Modern approaches to fair resource allocation as studied in collaborative decision making and computational social choice (see Chevaleyre *et al.* (2007)) provide a well-founded theoretical framework for such a research topic. A second topic for future research is the explainability of the results. Explanations are very important in order to convince decision makers and stakeholders to accept a solution or to give them indications of how to revise their preferences in order to obtain another solution. Explanation of optimality within combinatorial problems is a new research topic and Junker (2007) provides some first steps in this direction.

Acknowledgement

I would like to thank the anonymous reviewers for their very helpful remarks that substantially improved the quality of the paper.

References

- Nicolas Barnier, Pascal Brisset and Thomas Rivière. 2001. *Slot allocation with constraint programming: Models and results*. In International Air Traffic Management R&D Seminar ATM-2001.
- Dimitris Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni. 2008. *The air traffic flow management problem: An integer optimization approach*. Proceedings of IPCO, pages 34–46.
- Dimitris Bertsimas and Sarah Stock. 1994. *The air traffic flow management problem with enroute capacities*. MIT-report, MIT, Cambridge.
- Yann Chevaleyre, Ulle Endriss, Jérôme Lang, and Nicolas Maudet. 2007. *A Short Introduction to Computational Social Choice*. SOFSEM 2007: Theory and Practice of Computer Science, Springer, pages 51-69.
- CFMU. 2009. *Basic CFMU Handbook - General & CFMU Systems*. Eurocontrol CFMU, Brussels, 13.0 edition.
- M. Dalichampt, E. Petit, U. Junker, and J. Lebreton. 1997. *Innovative slot allocation*. Executive summary of EEC report no. 322, Eurocontrol Experimental Centre.
- Pierre Flener, Justin Pearson, Magnus Agren, Carlos Garcia-Avello, Mete Celiktin, and Soren Dissing. 2007. *Air-traffic complexity resolution in multi-sector planning*. Journal of Air Transport Management.
- William D. Harvey, Matthew L. Ginsberg. 1995. *Limited Discrepancy Search*. Proceedings of IJCAI, pages 607-615.
- Ulrich Junker. 2007. *Preference-Based Problem Solving for Constraint Programming*. CSCLP 2007, pages 109-126.
- Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. 1992. *Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling*. Artificial Intelligence, 58:161–205.
- Francesca Rossi, Peter van Beek, and Toby Walsh. 2006. *The Handbook of Constraint Programming*. Elsevier.
- Laurence A. Wolsey and George L. Nemhauser. 1999. *Integer and Combinatorial Optimization*. Wiley.