Contingency Plans for Air Traffic Flow and Capacity Management

Karl Sundequist Blomdahl, Pierre Flener, and Justin Pearson

Department of Information Technology
Uppsala University
Box 337, SE – 751 05 Uppsala, Sweden
Karl.Sundequist_Blomdahl.1559@student.uu.se, Pierre.Flener@it.uu.se (contact), Justin.Pearson@it.uu.se

***Abstract*—We present a constraint-based local search heuristic that contributes to solving the problem of generating contingency plans for air traffic flow and capacity management, which are to be used in the case of a catastrophic infrastructure failure within EUROCONTROL, the European Organisation for the Safety of Air Navigation. Experiments with the heuristic, implemented in *Comet*, on real-world flight plans for the entire European airspace show that it is feasible to automate the development of contingency plans, which is currently done by human experts. This is desirable as the development time goes down from two person months per year to a few CPU hours, and as it allows contingency plans to be generated with an increased frequency.**

***Index Terms*—Contingency planning, air traffic flow and capacity management, constraint programming, constraint-based local search, tabu search**

I. Introduction

A. Air Traffic Management

Air traffic management (ATM) is about managing and ensuring a safe, efficient, and fair flow of air traffic, assuming a negligible amount of side-effects, such as adverse weather conditions. During normal operation, the *Central Flow Management Unit* (CFMU) of the *European Organisation for the Safety of Air Navigation* (EUROCONTROL) uses several stages, each in increasing detail, to satisfy these three conflicting operational goals:

1) A strategic stage, taking place several months before the day of operation.
2) A pre-tactical stage that starts six days before the day of operation.
3) An online tactical stage during the day of operation. This stage is called the *air traffic flow and capacity management* (ATFCM) stage [1], and has two main functions:
   a) Calculate the demand of each airspace volume using live flight plan information.
   b) Adjust the number of allocated departure slots of the involved aerodromes, such that they optimise the objectives defined in the pre-tactical stage. These objectives typically include, but are not limited to, minimising the total flight delay and the air volume overload.

During an average day, the ATFCM unit handles approximately 30 000 flights spread over about 1 500 aerodromes.

This work has been financed by the European Organisation for the Safety or Air Navigation (EUROCONTROL) under its Care INO III programme (grant 08-121447-C). The content of the paper does not necessarily reflect the official position of EUROCONTROL on the matter.

B. Contingency Planning

This study focuses on the special case of an ATFCM failure due to any reason, such as downtime of the *computer-assisted slot allocation* (CASA) system. In such a situation, where no timely updates from ATFCM are available and the air controllers of each aerodrome have no idea whether it is proper to release a flight or not, a safe alternative is necessary. EUROCONTROL addresses this by a *contingency plan*, which contains a pre-defined number of allocated departure slots for each major aerodrome in such a way that certain safety and efficiency objectives are satisfied, for a maximum duration of one day. During the last twelve years, such a situation has occurred once, for a few hours. Nevertheless, EUROCONTROL requires the existence of such contingency plans, and they take time to develop.

An excerpt from such a contingency plan can be seen in Figure 1. It defines the number of departure slots that the aerodrome with the *International Civil Aviation Organisation* (ICAO) identifier EBBR (Brussels National Airport, Belgium) is allowed to release for each hour to various destination aerodromes. For example, from 09:00 to 12:00, a maximum of 7 flights are allowed to take off in the flow EBBR1, which is defined by the departure aerodrome EBBR and a destination aerodrome whose ICAO identifier starts with C (Canada), EG (Great Britain), EI (Ireland), K (United States), or M (Central America and Mexico). Similarly, only 4 flights whose departure and destination aerodrome match the description of the flow EBBR2 are allowed to take off per hour from 06:00 to 17:00.

The current contingency plan can always be downloaded from the CFMU website https://www.cfmu.eurocontrol.int/.

The generation of ATM contingency plans within the *EUROCONTROL Experimental Centre* (EEC) and the CFMU is currently done by two human experts (using a process described in Section II-B). They biannually (for the winter and summer timetables) develop a three-fold plan, namely one for weekdays, one for Saturdays, and one for Sundays.

Flow identifier	Flow description	Time span	Hourly rate
EBBR1	From: EBBR	00:00 – 06:00	2
	To: C EG EI K M	06:00 – 09:00	3
		09:00 – 12:00	7
		12:00 – 14:00	4
		14:00 – 22:00	8
		22:00 – 24:00	2
EBBR2	From: EBBR	00:00 – 06:00	1
	To: B EDDH EDDW EE EF EH EK EN ES	06:00 – 17:00	4
		17:00 – 21:00	6
		21:00 – 24:00	2

Figure 1. A contingency plan excerpt, which describes the hourly take-off rates of two flows originating from the aerodrome EBBR (Brussels National Airport, Belgium).

The total contingency planning time is two person-months per year, hence automated contingency planning is desirable. Another benefit with automating the process is that it could be done at the tactical level instead of the strategic level, which would increase the quality of the generated contingency plans.

C. Contributions and Organisation of this Paper

This paper presents a local search [2] heuristic that solves in just a few CPU hours the subproblem of finding the optimal hourly numbers of departure slots for *given* flows and time spans (which typically do not change much between contingency plans anyway) for the *entire* European airspace. It is intended as a feasibility study about replacing the human experts with constraint programming (CP) technology [3]. To our knowledge, this is the first time that contingency planning has been at least partially automated.

We here outline the model and the best of the two local search heuristics in our paper [5] at a specialist conference on CP, but with many more explanations about CP and much less technical detail about the model and the chosen heuristic.

The rest of this paper is split into four parts, dealing with the contingency planning problem in increasingly concrete terms: a formal definition of the problem as a constraint model (Section II), a local search heuristic that operates on the constraint model (Section III), experimental results with an implementation of the heuristic (Section IV), and a conclusion (Section V).

II. The Contingency Planning Problem

Informally, we address the following subproblem in contingency planning. We are given a set of flight plans and a set of flows with time spans. Our objective is to determine optimal hourly departure rates for these flows over these time spans, such that efficiency and safety of the global air traffic flow are optimal, under a fair allocation of departure slots. We measure efficiency as the total delay cost of all flights, under a first-submitted, first-served allocation. We measure safety as the total capacity overload cost of all air volumes. We minimise the weighted sum of these two terms.

We now give a formal description of this combinatorial optimisation problem as a constraint model, and give the current state of the art algorithm.

A. Constraint Model

Our constraint model is implemented in *Comet* [4], an object-oriented constraint programming language for the modelling of combinatorial problems. It has back-end solvers for (global) tree search interleaved with constraint propagation, for constraint-based local search, and for mixed integer linear programming. *Comet* is available at http://dynadec.com/.

Comet offers a very-high-level *modelling language* for fully declaratively specifying a combinatorial optimisation problem by (1) identifying the decisions that need to be made, namely the so-called *decision variables* (or *unknowns*) and their sets of possible values, called *domains*, (2) stating the *constraints* that are to be satisfied, and (3) defining the expression (called the *objective function*) that is to be minimised or maximised. Such a constraint model is (in principle) independent of the back-end solver.

An overview of our constraint model is given in Figure 2. The inputs are a set of flight plans, and the main decision variables denote the hourly rates of the output contingency plan. Through constraints that simulate the slot allocation process of air traffic control (ATC), the hourly rate decision variables are connected to overload decision variables and take-off delay decision variables, from which the safety and efficiency terms of the total cost are respectively determined. The fairness term of the total cost is obtained through the search heuristic rather than through constraints.

An instance of the contingency planning problem is defined by the following input and output data, where identifiers starting with capital letters denote given sets, subscripted identifiers denote constants, identifiers with indices within square brackets denote decision variables, identifiers that are Greek letters denote parameters, and all time moments are measured in seconds since some fixed origin:

- A set of flights $F = \{f_1, \ldots, f_m\}$, where each flight f_ℓ has a departure aerodrome $adep_\ell$, a destination aerodrome $ades_\ell$, an expected take-off time $etot_\ell$, a calculated take-off time $ctot[\ell]$, an expected landing time $eldt_\ell$, and a

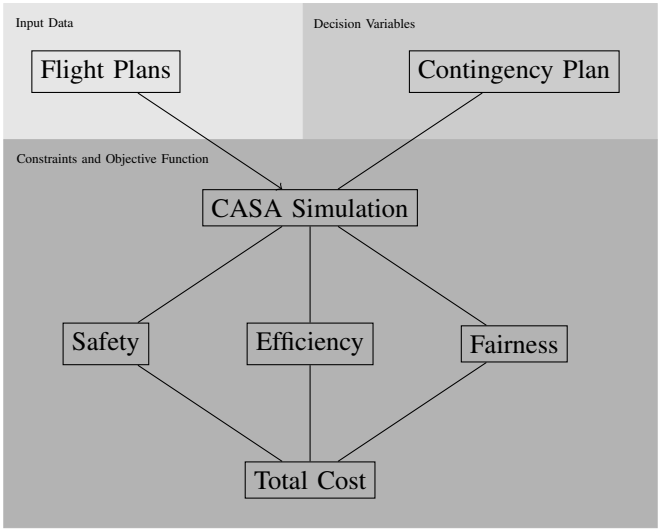

Figure 2. Overview of our constraint model.

take-off delay $delay[\ell]$. All later specified sets of flights are subsets of F.

- A set of air volumes $AV = \{av_1, \ldots, av_p\}$, where each air volume $av_a \in AV$ has a capacity cap_a that limits the hourly number of flights that can enter it for the duration dur_a. There is also a flight set $F_a \subseteq F$ for each air volume av_a that contains all flights that pass through av_a, where each flight $f_\ell \in F_a$ has an expected entering time $enter_{a,\ell}$ and a calculated entering time $cnter[a, \ell]$. In the real world, an air volume can represent either a part of the airspace or an aerodrome.
- A set of flows $\mathcal{F} = \{\mathcal{F}_1, \ldots, \mathcal{F}_n\}$, where each flow $\mathcal{F}_f$ consists of a set of flights F_f and a set of span-rate pairs $\mathcal{R}_f = \{r_1, \ldots, r_{o_f}\}$, where each span-rate pair r_i consists of a time span $span_i$ denoting when it is active, and an hourly rate of allocated departure slots $rate[i]$ in the integer interval $[1, demand_{f,i}]$, where $demand_{f,i}$ is the maximum number of flights that are planned to depart in flow $\mathcal{F}_f$ during the time span $span_i$:

$$demand_{f,i} = \max_{t \in T_i} |\{f_\ell \in F_f : ctot[\ell] \in [t, t + 3600)\}|$$

where set T_i contains the beginning times of all one-hour-long time intervals that fit inside the time span $span_i$ of the span-rate pair r_i, with a five minute step:

$$T_i = \{t \in span_i : t + 3600 \in span_i \wedge t \bmod 300 = 0\}$$

Further, for any two span-rate pairs r_i and r_j, where $i \neq j$, their spans must not overlap; however, the union of all spans does not need to be 00:00 – 24:00. There is also a flight set $F_f \subseteq F$ for each flow $\mathcal{F}_f$ that contains all flights matching the flow description. For example, Figure 1 defines two flows EBBR1 and EBBR2, where the flights are defined by a subset of F that matches the flow description, and the spans and rates are defined by the two right-most columns.

Additional decision variables used in the objective function will be defined in the following paragraphs.

Recall that ATM has three conflicting operational goals: ensure an efficient flow of air traffic (by minimising the total delay), ensure a safe flow of air traffic (by minimising the total capacity overload), and ensure a fair flow of air traffic. During a crisis situation, safety is especially important. Before giving the objective function, we first discuss the constraints induced by these operational goals.

1) Air Traffic Efficiency: The take-off delay $delay[\ell]$ of any flight f_ℓ is the difference between its calculated take-off time $ctot[\ell]$ and its expected take-off time $etot_\ell$:

$$delay[\ell] = ctot[\ell] - etot_\ell$$

where $ctot[\ell]$ is calculated using the allocated departure slots as defined by the span-rate pairs for each flow. These slots are assigned to flights using the first-submitted, first-served principle [6]. For example, consider the flow EBBR1 (defined in Figure 1), where there are three departure slots allocated for each hour between 06:00 and 09:00: if three flights with expected take-off times 06:00, 06:30, and 06:35 were available, then they would get the calculated take-off times 06:00, 06:40, and 07:00, and delays of 0, 600, and 1 500 seconds, respectively; note that no flight is given the 06:20 slot.

Similarly, the take-off delay $delay[\ell]$ of any flight f_ℓ also is the difference between its calculated entering time $cnter[a, \ell]$ into any air volume av_a and its expected entering time $enter_{a,\ell}$ into that air volume:

$$delay[\ell] = cnter[a, \ell] - enter_{a,\ell}$$

The *delay cost* of any flight f_ℓ is defined by a weight function, which was suggested to us by our research partners at the EEC:

$$delayCost[\ell] = \begin{cases} 1 & \text{if } 0 \text{ h} \leq delay[\ell] < 1 \text{ h} \\ 10 & \text{if } 1 \text{ h} \leq delay[\ell] < 2 \text{ h} \\ 20 & \text{if } 2 \text{ h} \leq delay[\ell] < 3 \text{ h} \\ 50 & \text{otherwise} \end{cases}$$

The weight scales exponentially because the real-world consequences do, in case of major disruption. For example, a flight with a low delay will probably only cause a slight interruption in the schedule, while a high delay might cause many flights to be cancelled.

The *total delay cost* is the sum of the delay costs of all the flights.

2) Air Traffic Safety: The *safety* of air traffic is determined by how crowded the air volumes are. The air volume av_a is capable of handling up to cap_a flights entering per hour, so any flight above this capacity creates an additional risk. Hence, safety is here defined by the amount that each air volume's hourly capacity is exceeded.

For each air volume av_a, a set T_a is defined that contains the beginning times of all one-hour-long time intervals that

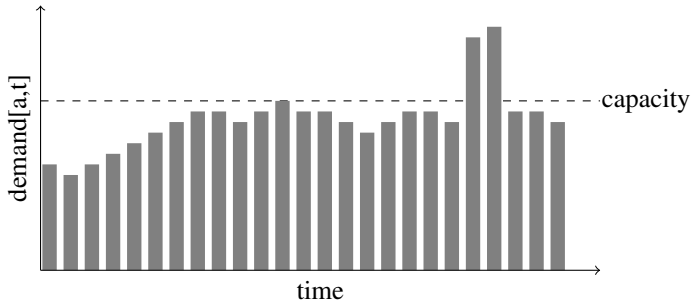

Figure 3. Demand for an air volume a over time: the vertical bars denote overlapping one-hour-intervals that start every five minutes, and the height of a bar for start time t indicates the number of flights scheduled to enter a during the hour following t, so that any excess of a bar over the capacity of a denotes a capacity overload.

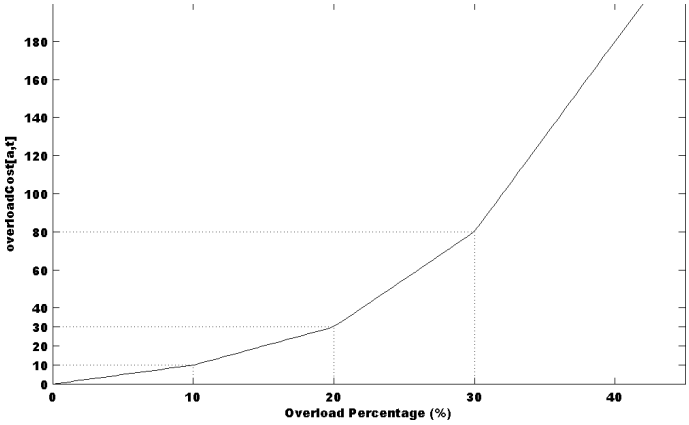

Figure 4. Piecewise linear function giving the capacity overload cost $overloadCost[a,t]$ in terms of the capacity overload percentage.

fit inside the air volume's capacity duration dur_a, with a five minute step, as depicted by the x axis in Figure 3:

$$T_a = \{t \in dur_a : t + 3600 \in dur_a \wedge t \bmod 300 = 0\}$$

The *capacity overload* of each air volume av_a and beginning time $t \in T_a$ is the number of flights, beyond the capacity of av_a, that enter av_a during the right-open time interval $[t, t + 3600)$:

$$overload[a,t] = \max(0, |\{f_\ell \in F_a : cnter[a,\ell] \in [t, t+3600)\}| - cap_a)$$

The *capacity overload cost* of air volume av_a and beginning time $t \in T_a$, denoted by $overloadCost[a,t]$, is defined by a piecewise linear function of the capacity overload percentage $\frac{overload[a,t]}{cap_a}$, where a suitable slope is defined for the overload percentage breakpoints 0%, 10%, 20%, and 30%. An illustration of the chosen function can be seen in Figure 4. Again, the cost scales exponentially, because a small capacity overload will likely only increase the workload of the affected ATM personnel slightly, while a large capacity overload might result in a mistake by the ATM personnel.

The *total capacity overload cost* is the sum of the capacity overload costs of all the air volumes and beginning times.

3) Air Traffic Fairness: The *fairness* of air traffic is here defined by how fairly the departure slots are allocated among the flows. No formal definition of fairness will be given at this point, as fairness is ensured by the search heuristic rather than by the constraint model, so we defer its discussion to Section III-D.

4) The Objective Function: The objective function, to be minimised, is a linear combination of the total delay cost and the total capacity overload cost, where α and β are parameters that can be chosen by the user:

$$\begin{aligned} cost &= \alpha \cdot \sum_{f_\ell \in F} delayCost[\ell] \\ &+ \beta \cdot \sum_{av_a \in AV} \sum_{t \in T_a} overloadCost[a,t] \end{aligned} \quad (1)$$

Experimental results and feedback from our research partners at the EEC suggest that $\alpha = 6$ and $\beta = 1$ are good values, because there are about six times fewer flights than air volumes and time steps. However, they can be changed to reflect any desired balance between a low delay and a low capacity overload.

B. Current State of the Art

The current state of the art, and the only known algorithm, to solve the contingency planning problem is the unpublished process used by the CFMU and EEC human experts. It has been described to us in the following high-level steps:

1) A statistical analysis is performed in order to point out the airspace volumes with a high demand. The duration and capacity of each air volume are recorded (there may be several durations per air volume).
2) An analysis of departing flows is made:
   - For the major European airports (i.e., with more than two arrivals or departures per hour on average), the traffic needs to be divided into main flows, where several destinations are grouped into each flow.
   - For the other airports, the flows are mainly divided into two categories: domestic flights and international flights. If the number of domestic flights is low, it seems better that a local flow manager handles this traffic.

Recall that it takes one person-month for two senior human experts to perform this algorithm, and that all this is done twice a year (once for the summer timetable and once for the winter timetable), for weekdays, Saturdays, and Sundays.

III. Local Search Heuristic

Comet [4] also offers a very-high-level *search language* for expressing a search procedure as well as its heuristics and meta-heuristics. The *Comet* constraint solving architecture takes care of all low-level, tedious, and error-prone computational details at run-time, and thereby significantly accelerates the development of effective and efficient search

procedures, as well as enormously eases the experimentation with alternative constraints or (meta-)heuristics. Often, this convenience is achieved at no additional cost in run-time compared to a hand-crafted program written in a low-level language. High-level *Comet* programs have even been reported to out-perform low-level programs that were hand-crafted by experts. Achievements based on *Comet* are listed at the *Comet* web-site and are easily found on the internet.

We here report on using the local search [2] back-end solver of *Comet*, which performs constraint-based local search (CBLS) [4]. This backend was chosen because of the sheer size of the data sets we have to handle. Trying the global search back-ends (tree search interleaved with propagation, and mixed integer linear programming) is considered future work.

In CBLS, constraints are used not only to state the problem but also to control the search. Search heuristics are guided by measures of constraint violation and variable violation. *Constraint violation* measures how close a constraint is to being satisfied. *Variable violation* measures for each decision variable in a constraint the variation of the constraint violation that could be achieved if that variable was suitably modified. Although these terms are not formally defined here, it is possible for a large number of constraints to come up with heuristically useful definitions of constraint and variable violations, and to compute them quickly.

Given an *initial assignment* of domain values to all the decision variables, a CBLS *heuristic* iteratively tries to find a better assignment that decreases the amount of constraint violation, by making a *move* to an assignment within the *neighbourhood* of the current assignment, that is a set of assignments that do not differ much from the current one. An assignment with zero (or minimal) constraint violation and an optimal value of the objective function is to be found. *Meta-heuristics* are used to escape local minima, that is when the neighbourhood contains no better assignments than the current one. Since the constraint and variable violations might thus need to be calculated thousands of times so as to pick the best move, and since thousands of moves might be needed, the algorithms and data structures implementing these violation calculations must be very efficient and, where possible, incremental.

We have developed two CBLS heuristics that operate on our constraint model. We here outline the better one of the two heuristics, which performs *tabu search*, and refer the reader to our paper [5] for a detailed description of the other heuristic, which performs *large neighbourhood search*.

The *generalised local search machine* (GLSM) of our tabu search heuristic can be seen in Figure 5. A GLSM [2] is a finite state machine that describes a local search heuristic by breaking it down into smaller algorithms, such that each state represents an individual algorithm and the edges represent the conditions for switching between these algorithms.

Our heuristic uses a tabu [7] meta-heuristic for escaping local minima. It uses a slightly modified objective function, which adds a penalty term to (1) in order to guide the heuristic toward a fair traffic flow, where $Penalty$ is a set of values

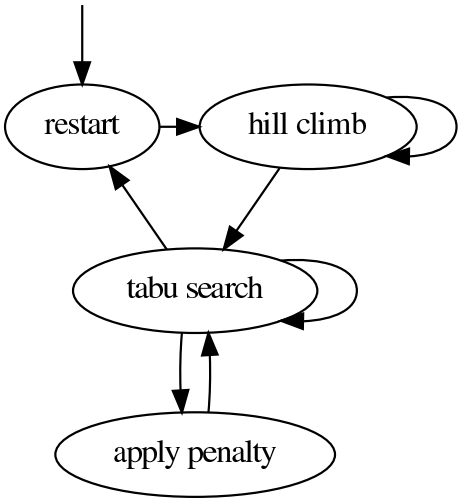

Figure 5. The generalised local search machine (GLSM) [2] of our tabu search heuristic.

maintained by integer invariants (discussed in Section III-D below):

$$\begin{aligned} cost &= \alpha \cdot \sum_{\ell \in F} delayCost[\ell] \\ &+ \beta \cdot \sum_{a \in AV} \sum_{t \in T_a} overloadCost[a,t] \\ &+ \sum_{p \in Penalty} p \end{aligned}$$

The heuristic can be summarised in the following steps, where each step and new terminology will be described in further details below:

1) (Re)start the search by assigning each flow rate variable $rate[i]$ a random value in its domain.
2) Hill-climb the current solution, until a local minimum has been reached.
3) Do a single run of tabu search, and then pick a random real number $u \in [0, 1]$. If $u < 0.05$, then pick a flow rate variable $rate[i]$ with an unfair value, add its penalty to the set $Penalty$, and repeat Step 3. Otherwise, if more than 200 iterations have gone by since the last improvement, then go to Step 1, else repeat Step 3.

The heuristic terminates once *maxIter* iterations have been completed, where *maxIter* is initialised to 1 000 and is set to the number of the current iteration plus 500 whenever a new best solution is found, unless this sum is smaller than 1 000.

The main source of diversification (directing the search toward another region of the search space) is Step 1, the main source of intensification (focussing the search on promising regions of the search space) is Step 2, while Step 3 performs a mix of both diversification and intensification.

A. The Restart Mechanism

The restart mechanism is the main source of diversification in our heuristic. It completely (re)starts the search by assigning each flow rate variable $rate[i]$ a random value in its domain. It

also clears the *tabu list*, which is the list of most recently visited assignments, stored for the sake of avoiding an untimely return to them.

B. Hill-climbing

The hill climbing algorithm is a *non*-greedy algorithm: during each iteration, it picks the *first* move $rate[i] := v$ such that the objective function is decreased. It does so until no such move can be found, that is until a local minimum has been reached. The method used to find this assignment is through the use of a meta-neighbourhood, which is a circular list of neighbourhoods $\{N_1, \ldots, N_q\}$ (where q is the total number of flow rate variables) that are searched in successive order until an improving assignment is found. Each neighbourhood N_i consists of all moves on flow rate variable $rate[i]$. The method terminates once a cycle has been completed with no improving assignment found.

C. Tabu Search

The tabu search [7] is the core of the heuristic. While it is the main contributor of neither intensification nor diversification, it ensures that the neighbourhood of a local minimum has been properly explored so that no improvements have been missed. During each iteration, it searches a neighbourhood (to be defined in the next paragraph) for a best non-taboo move $rate[i] := v$ and, after making the inverse move taboo for the number of iterations defined by the *tabu tenure*, it performs the assignment $rate[i] := v$. The only exception to this process is the *aspiration criterion*, which kicks in if the candidate solution is better than any solution found so far. If this is the case, then a move is performed even if it is in the tabu list. Our experiments were made with a tabu tenure $\tau = 8$.

The tabu search uses an asymmetrical stochastic neighbourhood that is designed to reduce the most severe air volume capacity overloads. It does so by finding the peak of each overload, and then picks one of these peaks to reduce at random, where the probability of each peak being picked is proportional to its value, hence higher peaks have a higher probability to be reduced. Once a peak has been determined, the neighbourhood consists of all moves on the flow rate variables $rate[i]$, where span-rate pair r_i corresponds to all flows $\mathcal{F}_f$ that contain a flight contributing to this peak (flights that cannot be anywhere else can be ignored).

D. Penalty Invariant

The apply-penalty state is the part of the heuristic that tries to ensure a high level of fairness of the air traffic flow. It does so by suitably modifying the value of the cost function under a fixed probability after each run of tabu search, such that the flow rate variable $rate[i]$ with the minimum $\frac{rate[i]}{demand_{f,i}}$ quotient is deemed *unfair* and an expression that tries to guide $rate[i]$ toward a fairer value is added to the set $Penalty$. It is an exponential expression that decreases the higher the value of $rate[i]$:

$$\gamma \cdot e^{-8 \cdot \frac{rate[i]}{demand_{f,i}}}$$

where γ is a user-definable parameter that controls how aggressively the heuristic should be guided toward fairness. In our experiments, we used $\gamma = 200$, which is only slightly aggressive.

IV. Experimental Results

Three real-life flight plans, which are comparable to those used by EUROCONTROL when generating the official contingency plans, have been provided by the EEC, and have been used as *training* flight plans:

- A weekday (Friday 2008-06-27), with 261 flows (320 rates), 36 161 flights, and 348 air volumes.
- A Saturday (2008-08-30), with 256 flows (387 rates), 29 842 flights, and 348 air volumes.
- A Sunday (2008-08-31), with 259 flows (397 rates), 31 024 flights, and 348 air volumes.

When translated into a constrained optimisation problem, each instance yields approximately 150 000 constraints and 50 000 decision variables.

All experiments were done on a Linux x86-64 dual-core laptop with 4GB of primary memory, 2MB of L2 cache, and a CPU frequency of 2.2GHz. Under *Comet* version 2.0.1, the tabu search usually terminated after approximately three CPU hours.

Our own comparison between contingency plans generated by our heuristic and contingency plans generated by the EEC and CFMU human experts (denoted by EEC) can be seen in Figure 6, giving the expected take-off delay (in seconds, only for the *delayed* flights) and the 95th percentile thereof, as well as the expected air volume capacity overload percentage (only for the *overloaded* sectors) and the 95th percentile thereof. We observe that our heuristic outperforms the EEC algorithm on both measures.

This good performance of our heuristic has been validated independently by the EEC and CFMU human experts, using their internal simulation tool COSAAC. They compared our contingency plans and their contingency plans on realistic *test* flight plans (which were not given to us), though *not* according to the objective function we used during our optimisation, but more realistically according to a CASA-style slot allocation, as if CASA was actually *not* down. Indeed, our objective function and constraints only *simulate* (our understanding of) CASA, as calls to CASA itself for every candidate move in the neighbourhood of every iteration of local search would be prohibitively expensive.

Figures 7 and 8 respectively give the average take-off delay (in seconds, only for the *delayed* flights) and the average capacity overload percentage (only for the overloaded air volumes), on a weekday, a Saturday, and a Sunday in the European summer 2008 timetable according to the contingency plans generated by the algorithm of the EUROCONTROL Experimental Centre (EEC), our tabu search heuristic (tabu), and our large neighbourhood heuristic (LNS). We observe that our heuristic significantly decreases both the average take-off delay (among the delayed flights) and the capacity overload (among

Contingency Plan	E(delay)	p_{95}(delay)	E(overload)	p_{95}(overload)
EEC 2008-06-27	645.6 sec	2340.0 sec	29%	100%
Tabu 2008-06-27	310.2 sec	1200.0 sec	27%	72%
EEC 2008-08-30	528.1 sec	1800.0 sec	23%	61%
Tabu 2008-08-30	316.1 sec	1200.0 sec	22%	56%
EEC 2008-08-31	407.0 sec	1500.0 sec	29%	68%
Tabu 2008-08-31	345.9 sec	1264.5 sec	24%	57%

Figure 6. Our analysis: Expected take-off delay (in seconds, only for the *delayed* flights) and the 95th percentile thereof, as well as the expected air volume capacity overload percentage (only for the *overloaded* air volumes) and the 95th percentile thereof, on a weekday, a Saturday, and a Sunday in the European summer 2008 timetable according to the contingency plans generated by the algorithm of the EUROCONTROL Experimental Centre (EEC) and our tabu search heuristic (tabu).

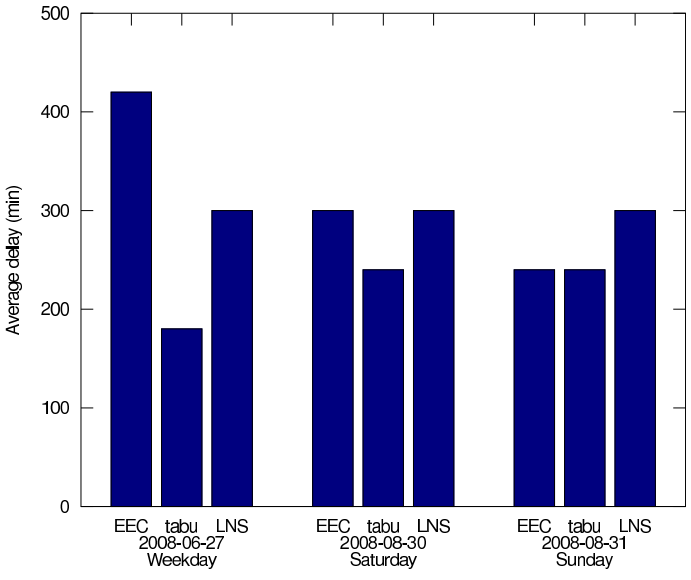

Figure 7. EEC/CFMU analysis: Average take-off delay (in seconds), only for the *delayed* flights, on a weekday, a Saturday, and a Sunday in the European summer 2008 timetable according to the contingency plans generated by the algorithm of the EUROCONTROL Experimental Centre (EEC), our tabu search heuristic (tabu), and our large neighbourhood heuristic (LNS).

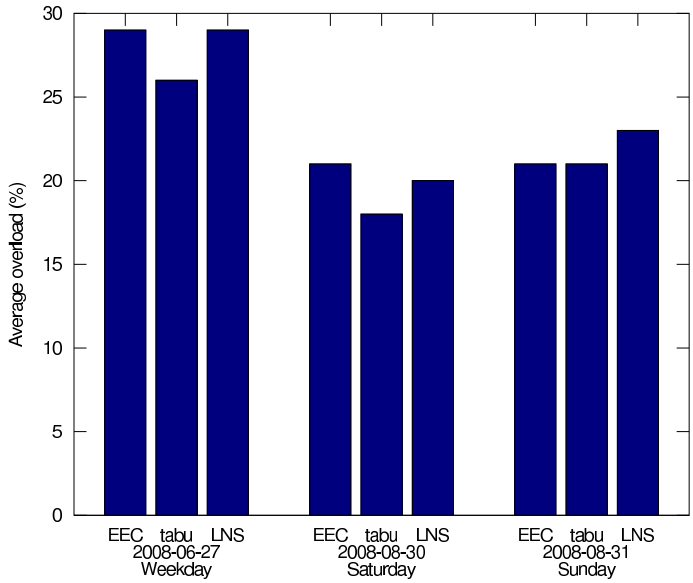

Figure 8. EEC/CFMU analysis: Average capacity overload percentage, only for the *overloaded* air volumes, on a weekday, a Saturday, and a Sunday in the European summer 2008 timetable according to the contingency plans generated by the algorithm of the EUROCONTROL Experimental Centre (EEC), our tabu search heuristic (tabu), and our large neighbourhood heuristic (LNS).

overloaded air volumes) of the contingency plans generated by the human experts.

Our tabu search heuristic not only decreases the take-off delay of delayed flights (as seen in Figure 7) but also delays significantly fewer flights compared to the EEC algorithm (26% vs 35% on the chosen weekday, 31% vs 38% on the chosen Saturday, and 31% vs 35% on the chosen Sunday).

Finally, our tabu search heuristic not only decreases the capacity overload of overloaded air volumes (as seen in Figure 8) but also overloads fewer air volumes compared to the EEC algorithm (27% vs 28% on the chosen weekday, 30% vs 31% on the chosen Saturday, and 26% vs 28% on the chosen Sunday).

V. Conclusion

This work is part of a feasibility study about whether it is possible to automate the development of contingency plans for EUROCONTROL, the *European Organisation for the Safety of Air Navigation*. Our positive results were expected, due to the similarities between the contingency planning problem and scheduling problems, which have been solved successfully using constraint programming technology for a couple decades. It thus seems to be possible to automate contingency planning efficiently enough with constraint programming technology.

Regarding future work, recall that this paper addresses the subproblem of finding the optimal hourly departure rates for predefined flows and time spans. The latter have been produced by human experts, and do actually then not change much from one year to another. However, this dependency on predefined flows and time spans must be eliminated. Currently, this is our most important issue. Ideally, the search for an optimal set of flows and time spans would be integrated into our heuristic.

Acknowledgements

We thank Serge Manchon, Elisabeth Petit, and Bernard Kerstenne at the EUROCONTROL Experimental Centre for their feedback on our progress. Many thanks also to the anonymous referees for their useful suggestions.

REFERENCES

[1] EUROCONTROL, *Air Traffic Flow & Capacity Management Users Manual*, 14th ed. EUROCONTROL CFMU, March 2010, available at http://www.cfmu.eurocontrol.int/j_nip/cfmu/public/standard_page/library_handbook_supplements.html.

[2] H. H. Hoos and T. Stützle, *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.

[3] F. Rossi, P. van Beek, and T. Walsh, Eds., *Handbook of Constraint Programming*. Elsevier, 2006.

[4] P. Van Hentenryck and L. Michel, *Constraint-Based Local Search*. The MIT Press, 2005.

[5] K. Sundequist Blomdahl, P. Flener, and J. Pearson, "Contingency plans for air traffic management," in *Proceedings of CP'10, the 16th international conference on Constraint Programming*, ser. Lecture Notes in Computer Science, vol. 6308, D. Cohen, Ed. Springer-Verlag, 2010, pp. 643–657.

[6] EUROCONTROL, *General & CFMU Systems*, 14th ed. EUROCONTROL CFMU, March 2010, available at http://www.cfmu.eurocontrol.int/j_nip/cfmu/public/standard_page/library_handbook_supplements.html.

[7] F. Glover and M. Laguna, "Tabu search," in *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, 1993, pp. 70–150.