The Path&Cycle formulation for the Hotspot Problem in Air Traffic Management

Carlo Mannino
SINTEF
[Forskningsveien 1, Oslo, Norway]
carlo.mannino@sintef.no

Giorgio Sartor
SINTEF
[Forskningsveien 1, Oslo, Norway]
giorgio.sartor@sintef.no

Abstract

The Hotspot Problem in Air Traffic Management consists of optimally rescheduling a set of airplanes that are forecast to occupy an overcrowded region of the airspace, should they follow their original schedule. We first provide a MILP model for the Hotspot Problem using a standard big-M formulation. Then, we present a novel MILP model that gets rid of the big-M coefficients. The new formulation contains only simple combinatorial constraints, corresponding to paths and cycles in an associated disjunctive graph. We report computational results on a set of randomly generated instances. In the experiments, the new formulation consistently outperforms the big-M formulation, both in terms of running times and number of branching nodes.

2012 ACM Subject Classification Applied computing → Transportation

Keywords and phrases Air Traffic Management, Hotspot Problem, Job-shop scheduling, Mixed Integer Linear Programming

Digital Object Identifier 10.4230/OASIcs.ATMOS.2018.14

Funding Project OPSTRA (Nr. 267554/080) founded by the Norwegian Research Council

1 Introduction

An important task in Air Traffic Management is the dynamic (re)scheduling of flights in order to preemptively avoid that regions of the airspace would become overcrowded at some point in time after the flights have departed (the frequency at which this scheduling happens is not important in this paper). This is necessary to avoid overburdened air traffic controllers. In fact, in order to guarantee the safety of air travel in large regions, the airspace is partitioned into small volumes called *control sectors*. At any time, each such sector is managed by one or more air traffic controllers. Due to safety reasons, each controller can only watch up to a certain number of airplanes. The maximum number of airplanes controllable in a given sector is called *capacity* (of the sector). If too many airplanes occupy a sector at a given time, then there is an *hotspot* (see, e.g. [1, 2]). Hotspots can be avoided by delaying some flights, holding airplanes on the ground. Our objective is to compute a hotspot-free schedule for a set of airplanes in a large region of the airspace while minimizing the total delay.

For our purposes, it suffices to describe the route of each airplane as an ordered sequence of sectors, starting at the departure airport and ending at the arrival airport. Even if airplanes can adjust their cruising speed to a certain extent, in this paper we assume that such speed is fixed, which implies that the time to traverse a given sector is also fixed. This is in accordance with the standard subdivision of roles in air traffic management. In fact, the speed of an airplane is monitored and possibly adjusted usually only by an air traffic controller within his/her control sector. Instead, the schedule of a flight is assessed and possibly recomputed many hours ahead its original departure time from a central authority (e.g. in Europe, Eurocontrol). This procedure takes into consideration the official timetable for all the flights traversing a region of the airspace and their associated routes. Already this timetable can contain one or more hotspots. More typically, hotspots may emerge because of some unpredicted event, such as a sudden delay in one or more aircraft ground operations, bad weather conditions, or even the reduction of the capacity of one or more sectors.

When a hotspot is predicted, the authorities are required to implement some actions to eliminate it. These actions generally consist of delaying the departure of some of the airplanes. So, a natural problem arises: which airplanes should be held at the departure airport, and for how long? Clearly we would like to minimize a measure of the overall delay that is introduced with these actions. We call this the *Hotspot Problem (HP)*.

A few recent papers address variants of the hotspot problem. In [6], the airspace is subdivided into micro-cells of unit capacity, and airplanes can be delayed at the departure, but only within the assigned time slot. A related problem, but on the side of the airlines rather than of the controlling bodies, is addressed in [7]. Here, the authors assume that, in order to mitigate congestion, the control authority issues a number of flight restrictions (FCA) within feasible time slots for the flights of some airlines. The airline is then confronted with the decision of how to modify flights trajectories in order to satisfy the FCAs. The feasible trajectories are chosen in a predefined, finite set. Finally, in [5] an overarching, time-indexed formulation is developed for a problem which includes, as subproblem, capacity requirements in certain points in space.

All the above mentioned papers focus on modeling issues, using either constraint (CP) or mixed integer (MIP) programming. The resulting formulations are then solved by invoking a state-of-the-art CP or MIP solver. However, in our experience, this approach typically does not suffice to tackle instances of practical size. Indeed, the standard formulations for this kind of problems are the big-M and the time-indexed formulations. The former usually provides weak bounds, and thus large search trees; the latter tends to grow to intractable dimensions very quickly. In this paper we instead develop a new MILP formulation for the Hotspot Problem that allows us to significantly improve over a standard big-M formulation.

2 A MILP big-M model for the Hotspot Problem

We start by introducing a standard big-M model for the Hotspot Problem. It extends the model for job-shop scheduling problem with blocking and no-wait constraints introduced in [4] and exploited in several papers for different transportation problems.

The Hotspot Problem is characterized by a set of sectors S (i.e., the airspace) and a set of flights F. Each sector $s \in S$ is associated with a maximum capacity c_s. A *route node* is a pair (f, s), where $f \in F$ is a flight and $s \in S$ is a sector. For each flight $f \in F$, we define its flight route as an ordered sequence of route nodes: $\big((f, s_1), (f, s_2), \dots, (f, s_q)\big)$ where s_1, s_q are the sectors in which the departure and arrival airports are located, respectively, and s_{i-1}, s_i are adjacent sectors, for $i = 2, \dots, q$. With some abuse of notation, we denote by $(f, s{+}1)$ the route node that immediately follows (f, s) in the flight route of f.

Let R be the set of all route nodes for all flights in F, D the set of all departure nodes, and A the set of all arrival nodes. With each route node $(f, s) \in R$ we associate the fixed time $\Lambda_{(f,s)}$, that is the time flight f takes to traverse sector s. This time can be obtained from the the official flight schedule and mainly depends on the entry and exit point of the airplane in that sector. Moreover, we indicate with Γ_f the minimum departure time of a flight f in respect to a certain reference time that is common to all $f \in F$.

We can now start building the MILP model by associating a scheduling variable $t_{(f,s)} \in \mathbb{R}$ to each route node $(f,s) \in R$, where $t_{(f,s)}$ represents the time flight f enters sector s. Note that the time a flight exits a particular sector is equal to the time the flight enters the subsequent sector in its route. We also introduce a fictitious variable $t_o \in \mathbb{R}$, which serves as a reference time for all airplanes. Thus, we have

$$t_{(f,s)} - t_o \geq \Gamma_f, \quad (f,s) \in D. \tag{1}$$

Now let $(f,s), (f,s{+}1) \in R$ be two consecutive route nodes in a particular flight route. Then the following *precedence constraints* must hold:

$$t_{(f,s+1)} - t_{(f,s)} = \Lambda_{(f,s)}. \tag{2}$$

In fact, we assume that each airplane travels at fixed speed throughout its route, but it is allowed to delay its departure.

Now, for each pair of distinct flights $f, g \in F$, we denote by $S(f,g)$ the shared sectors, and for each $s \in S(f,g)$ we introduce the binary quantity x^s_{fg}, which is 1 if and only if f and g meet in s. Consider now a set of distinct flights $\bar{F} \subseteq F$ traversing a sector s, and assume that $|\bar{F}| > c_s$. Then, the following *hotspot constraints* must hold:

$$\sum_{\{f,g\} \subseteq K} x^s_{fg} \leq \binom{c_s + 1}{2} - 1, \quad K \subseteq \bar{F}, |K| = c_s + 1, s \in S. \tag{3}$$

It is easy to see that these constraints are enough to guarantee that at most c_s flights meet in sector s. This is indeed a straightforward application of the well-known *Helly's Theorem* in one dimension, which states that a set of intervals in $\mathbb{R}$ (i.e., the time) has a nonempty intersection if and only if every pair intersects.

Observe that, for a pair of distinct flights f, g traversing a sector s, exactly one of the following three conditions must occur: a) flight f and g meet in sector s, or b) flight f traverses sector s before flight g, or c) flight g traverses sector s before flight f. For each ordered pair of flights $(f,g) \in \bar{F}$, we define y^s_{fg} to be equal to 1 if f exits s before g enters, and 0 otherwise. Then, we have that

$$y^s_{fg} + y^s_{gf} + x^s_{fg} = 1, \quad \{f,g\} \subseteq \bar{F}, s \in S. \tag{4}$$

So, precisely one of the above three variables will be 1 in any feasible solution. Accordingly, for every $\{f,g\} \subseteq \bar{F}, s \in S$, the schedule t will satisfy a family of *disjunctive constraints* that can be modeled by means of a conjunction of big-M constraints as follows:

$$\begin{aligned}
&(i) && t_{(g,s)} - t_{(f,s+1)} \geq -M(1 - y^s_{fg}),\\
&(ii) && t_{(f,s)} - t_{(g,s+1)} \geq -M(1 - y^s_{gf}),\\
&(iii) && t_{(g,s+1)} - t_{(f,s)} \geq -M(1 - x^s_{fg}),\\
&(iv) && t_{(f,s+1)} - t_{(g,s)} \geq -M(1 - x^s_{fg}),\\
& && y^s_{fg}, y^s_{gf}, x^s_{fg} \in \{0,1\},
\end{aligned} \tag{5}$$

where M is a suitably large positive constant, and $t_{(h,s+1)}$ is the time the flight h enters the sector next to s in its route (i.e., the time h exits sector s). Indeed, if $y^s_{fg} = 1$ then (ii) and (iii) and (iv) become redundant, whereas constraint (i) reduces to $t_{(g,s)} - t_{(f,s+1)} \geq 0$, which implies that f exits s before g enters s. Similarly, when (ii) is active, g exits s before f enters. On the other hand, when $x^s_{fg} = 1$, then (i) and (ii) become redundant, whereas (iii) and (iv) are active, implying that both f and g exit the sector s after the other flight enters it (i.e., they meet in s).

In conclusion, a complete MILP formulation for the Hotspot Problem can be obtained by considering constraints (1) and (2) for all routes, and constraints (3), (4), and (5) for all sectors $s \in S$ and all sets $\bar{F} \subseteq F$ of flights exceeding the capacity c_s of s. We call this the big-M formulation (BF).

Let $P \subset \mathbb{R}^p$ be the set of points (y, x, t) satisfying all such inequalities, then our problem reduces to $\{\min c(t) : (y, x, t) \in P\}$.

The objective $c(t)$ may vary from instance to instance, but in this paper it will simply be the (weighted) delay at destination.

In principle, formulation BF could be solved by resorting to any off-the-shelf MILP solver. However, the families of constraints (3), (4) and (5) can grow very quickly[1], making the formulation impractical even for small-medium size realistic instances. A standard way to tackle this issue is to make use of row generation. Namely, constraints are generated dynamically and added to the model only if they are violated by the incumbent integer feasible solution. An algorithm to solve formulation BF is presented in Algorithm 1.

Algorithm 1 An algorithm for the big-M formulation

```
$\mathcal{P} \leftarrow$ Set of precedence constraints
$\mathcal{H} \leftarrow \emptyset$                                              ▷ Set of hotspot constraints
$\mathcal{D} \leftarrow \emptyset$                                              ▷ Set of disjunctive constraints
$\mathcal{M} \leftarrow \min_{y,x,t} c(t)$, subject to $\mathcal{P}$,$\mathcal{H}$, and $\mathcal{D}$          ▷ MILP model for $BF$
$(y, x, t) \leftarrow$ incumbent solution of $\mathcal{M}$
while true do
    Solve $\mathcal{M}$
    if $y, x, t$ violates a disjunction constraint $D$ then
        $\mathcal{D} \leftarrow \mathcal{D} \cup D$                                   ▷ Row generation
        continue
    else if $y, x, t$ violates a hotspot constraint $H$ then
        $\mathcal{H} \leftarrow \mathcal{H} \cup H$                                   ▷ Row generation
        continue
    else
        break                                                  ▷ Found optimal!
```

3 A non-compact reformulation

In the previous section we presented a compact formulation that fully characterizes the Hotspot Problem, and we presented an algorithm to solve it in a practical context. However, BF still contains one major sources of complexity. In fact, in order to make the constraints in

[1] The total number of constraints is $O(|S||F|^{c_s})$ in (3), and is $O(|S| \times |F|^2)$ in (4) and (5).

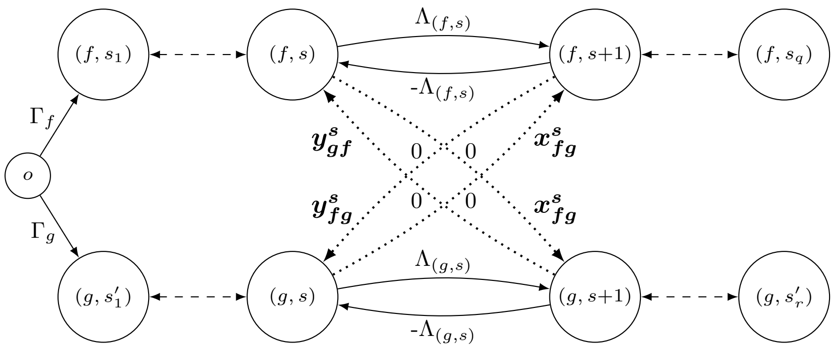

Figure 1 A disjunctive graph for a pair of flights f, g that meet in sector s. Note that, the sector associated with the node $(f, s{+}1)$ might be different from the sector associated with $(g, s{+}1)$. For each flight, the *precedence edges* enforce a fixed traversing time Λ in the corresponding sector. Instead, the zero-weighted *conflict edges* are each associated to a binary variable, and they become binding only if the corresponding binary variable is equal to 1.

(5) redundant for certain values of the binary variables, we made use in BF of the (in)famous big-M method. Unfortunately, including a large coefficient in the model usually makes the formulation weak and prone to return poor bounds in the search trees, often leading to slow solution times.

Our approach to tackle this problem and solve $\{\min c(t) : (y, x, t) \in P\}$ extends the methodology first developed in [3]. In particular, we exploit a Benders-like decomposition to obtain a (master) problem only in the binary variables, plus a few continuous variables to represent the objective function. The decomposition allows us to get rid of big-M coefficients (at the cost of an increased number of linear constraints). Moreover, the constraints of the reformulated master correspond to basic graph structures in the so called disjunctive graph, such as paths and cycles.

We sketch here how the reformulation is obtained. First, we consider the disjunctive graph associated with our big-M formulation BF. This is a directed graph $G = (V, E)$ obtained by considering a vertex for every route node $u \in R$, plus an extra node: the origin o. A directed edge (u, v) of length l_{uv} in the disjunctive graph represents an inequality $t_v - t_u \geq l_{uv}$, indicating that the minimum travel time from route node u to route node v is l_{uv}. Therefore, we can add edges to G to represent some of the constraints of BF.

In particular, the origin is connected with a direct edge (o, d_f) to the node $d_f \in D$, corresponding to the departure node of flight $f \in F$. The length of edge (o, d_f) equals the minimum departure time of flight $f \in F$, Γ_f. Then we add an edge (u, v) of weight Λ_u and an edge (v, u) of weight $-\Lambda_u$, for every constraint (2). These are called the *precedence edges*.

Consider now inequalities (5.i)-(5.iv). For every variable y^s_{fg}, we add the edge (u, v) with length zero, where u, v are the route nodes associated with $t_{(f,s+1)}, t_{(g,s)}$, respectively. In fact, if $y^s_{fg} = 1$ then $t_{(g,s)} - t_{(f,s+1)} \geq 0$. These edges are called *y-edges*, and the set of y-edges is denoted by K_y. Similarly, with every variable x^s_{fg} we associate two edges of length zero: these edges are called *x-edges*, and the set of x-edges is denoted by K_x. For $e \in K_y$ ($e \in K_x$), we let y_e (x_e) be the y-variable (x-variable) that generates e. The set $K_y \cup K_x$ is the set of *conflict edges*.

Figure 1 shows how a disjunctive graph would look like for a couple of flights that meet at least in one sector.

Consider now a feasible solution $(\bar{y}, \bar{x}, \bar{t})$ to (2), (3), (4) and (5). Let $G(\bar{y}, \bar{x})$ be the graph obtained from the disjunctive graph by removing all the edges $e \in K_y$ with $\bar{y}_e = 0$ and all the edges $e \in K_x$ with $\bar{x}_e = 0$. Note that the vector $(\bar{y}, \bar{x}) \in \{0,1\}^{K_y \cup K_x}$ is the incidence vector of the subset of conflict edges contained in $G(\bar{y}, \bar{x})$, and we say that such edges are *selected* by $(\bar{y}, \bar{x})$. Then the following lemma holds.

▶ **Lemma 1.** *i.) $G(\bar{y}, \bar{x})$ does not contain strictly positive directed cycles. ii.) If $(\bar{y}, \bar{x}, \bar{t})$ is an optimal solution, and $\bar{t}_{a_f}$ is the associated arrival time of flight $f \in F$, then $\bar{t}_{a_f}$ equals the length of the longest path from the origin o to route node $a_f \in A$ in $G(\bar{y}, \bar{x})$.*

Proof. When variables y, x are fixed, it is easy to see that the problem BF reduces to the dual of a max-cost flow problem. Then, the result follows immediately from well-known theorems of network theory. ◀

Note that our objective function is simply $c(t) = \sum_{a \in A} t_a$, but the following results can be immediately extended to any function non-decreasing with t.

The lemma has two straightforward consequences: any feasible solution corresponds to a selection $\bar{y}, \bar{x}$ of conflict edges such that $G(\bar{y}, \bar{x})$ does not contain a strictly positive directed cycle; and, for any feasible selection $\bar{y}, \bar{x}$, the best possible scheduling corresponds to the longest path tree in $G(\bar{y}, \bar{x})$.

In this context, the Hotspot Problem (HP) can be stated as follows: *find a feasible selection y, x of conflict edges such that $G(y, x)$ does not contain a strictly positive directed cycle, the sum of the lengths of the longest paths from the origin o to the arrival nodes $a \in A$ is minimum, and the resulting schedule is hotspot-free.*

Let us denote by $\mathcal{C}$ the set of strictly positive length di-cycles of G, and by $L^*(y, x, u)$ the length of the longest path from o to u in $G(y, x)$. Then a new formulation for the Hotspot Problem can be written as follows:

$$\begin{array}{lll}
\min & \sum_{u \in A} L^*(y, x, u) & \\
s.t. & & \\
(i) & y^s_{fg} + y^s_{gf} + x^s_{fg} = 1, & \{f, g\} \in F, s \in S, \\
(ii) & \sum_{e \in C \cap K_y} y_e + \sum_{e \in C \cap K_x} x_e \leq |C \cap K| - 1, & C \in \mathcal{C}, \\
(iii) & \sum_{\{f,g\} \subseteq \bar{F}} x^s_{fg} \leq \binom{|\bar{F}|}{2} - 1, & s \in S, \bar{F} \subseteq F, |\bar{F}| = c_s + 1, \\
 & y \in \{0,1\}^{|K_y|}, x \in \{0,1\}^{|K_x|}. &
\end{array} \tag{6}$$

Constraint (6.ii) ensures that one does not select all the conflict edges contained in a strictly positive di-cycle. Equivalently we write

$$\begin{array}{lll}
\min & \sum_{u \in A} \mu_u & \\
s.t. & & \\
(i) & y^s_{fg} + y^s_{gf} + x^s_{fg} = 1, & \{f, g\} \in F, s \in S, \\
(ii) & \sum_{e \in C \cap K_y} y_e + \sum_{e \in C \cap K_x} x_e \leq |C \cap K| - 1, & C \in \mathcal{C}, \\
(iii) & \sum_{\{f,g\} \subseteq \bar{F}} x^s_{fg} \leq \binom{|\bar{F}|}{2} - 1, & s \in S, \bar{F} \subseteq F, |\bar{F}| = c_s + 1, \\
(iv) & \mu_u \geq L^*(u, y, x), & u \in A, \\
 & y \in \{0,1\}^{|K_y|}, x \in \{0,1\}^{|K_x|}, \mu \in \mathbb{R}^{|A|}. &
\end{array} \tag{7}$$

We can finally rewrite constraints (7.*iv*) in a way that can be immediately exploited in a row generation algorithm. We denote by $\mathcal{H}$ the set of all $G(y,x)$ for (y,x) satisfying (7.*i*), (7.*ii*), (7.*iii*). If $H \in \mathcal{H}$, then we denote by $P_u(H)$ the (set of edges of a) longest path from o to u in H, and by $L_u(H)$ the length of $P_u(H)$. The final reformulation can now be written as follows:

$$
\begin{array}{lll}
\min & \sum\limits_{u\in A} \mu_u & \\
s.t. & & \\
(i) & y^s_{fg} + y^s_{gf} + x^s_{fg} = 1, & \{f,g\} \in F, s \in S, \\
(ii) & \sum\limits_{e\in C\cap K_y} y_e + \sum\limits_{e\in C\cap K_x} x_e \le |C \cap K| - 1, & C \in \mathcal{C}, \\
(iii) & \sum\limits_{\{f,g\}\subseteq \bar F} x^s_{fg} \le \binom{|\bar F|}{2} - 1, & s \in S, \bar F \subseteq F, |\bar F| = c_s + 1, \\
(iv) & L_u(H)(\sum\limits_{e\in K_y\cap P_u(H)} y_e + \sum\limits_{e\in K_x\cap P_u(H)} x_e - |K \cap P_u(H)| + 1) \le \mu_u, & u \in A, H \in \mathcal{H}, \\
 & y \in \{0,1\}^{|K_y|}, x \in \{0,1\}^{|K_x|}, \mu \in \mathbb{R}^{|A|}. &
\end{array}
\tag{8}
$$

Indeed, consider a feasible solution $(\bar y, \bar x, \bar\mu)$ to (8). Let $\bar H = G(\bar y, \bar x)$ and let $\bar P_u(\bar H)$ be a longest path from o to u in $\bar H$. Then all conflict edges on $\bar P_u(\bar H)$ are selected by $\bar y, \bar x$ and we have

$$\sum_{e\in K_y\cap \bar P_u(\bar H)} \bar y_e + \sum_{e\in K_x\cap \bar P_u(\bar H)} \bar x_e - |K \cap \bar P_u(\bar H)| + 1 = 1$$

which in turn implies

$$\bar\mu_u \ge L_u(\bar H) = L^*(\bar y, \bar x, u).$$

On the other hand, when one or more edges in a path are not selected, then the constraint is satisfied for any $\mu_u \ge 0$.

We call Problem (8) the *Path&Cycle* formulation (*PC*) of the Hotpot Problem and we solve it with the algorithm described in Algorithm 2. Constraints (8.i) are called the *selection constraints*, (8.ii) are called the *cycle constraints*, (8.iv) are called the *path constraints*, and (8.iii) are called the *hotspot constraints.*

In short, the algorithm works by generating combinations of the x, y variables such that $\sum_{u\in A} \mu_u$ is minimized. If a particular solution $(\bar y, \bar x)$ is such that $G(\bar y, \bar x)$ contains a positive cycle or (8.1) is not satisfied, then the corresponding constraint is added the problem. Otherwise, the longest paths $L_u(G(\bar y, \bar x)), u \in A$ are computed. If there exists a $u \in A$ such that $L_u(G(\bar y, \bar x)) > \mu_u$, then the corresponding path constraint is added to problem. Otherwise, the algorithm is able to use variables x, y, μ to produce a schedule for the t variables. If this schedule violates the capacity of any of the sectors, then the corresponding hotspot constraint is added to the problem. Finally, if none of these inequalities needs to be added, then we found the optimal solution.

4 Computational experiments

In this section we analyze the performance of the *BF* formulation versus the *PC* formulation on randomly generated instances. Each instance represents a snapshot (in time) of the

Algorithm 2 An algorithm for the *Path&Cycle* formulation

```
$G \leftarrow$ disjunctive graph
$\mathcal{S} \leftarrow \emptyset, \mathcal{H} \leftarrow \emptyset$                    ▷ Sets of selection and hotspot constraints
$\mathcal{C} \leftarrow \emptyset, \mathcal{P} \leftarrow \emptyset$                    ▷ Sets of cycle and path constraints
$\mathcal{M} \leftarrow \min_{y,x,\mu} \mu^T \mathbb{1}$, subject to $\mathcal{S}$,$\mathcal{H}$,$\mathcal{C}$, and $\mathcal{P}$     ▷ MILP model for $PC$
$(y, x, \mu) \leftarrow$ incumbent solution of $\mathcal{M}$
while true do
    while true do
        Solve $\mathcal{M}$
        if $y, x$ violates a selection constraint $S$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup S$                          ▷ Row generation
            continue
        else if $y, x$ violates a cycle constraint $C$ then
            $\mathcal{C} \leftarrow \mathcal{C} \cup C$                          ▷ Row generation
            continue
        else
            Find the longest paths in $G(y, x)$
            if $(y, x, \mu)$ violate a path constraint $P$ then
                $\mathcal{P} \leftarrow \mathcal{P} \cup P$                      ▷ Row generation
                continue
            else
                break
    if $x, y$ violates a hotspot constraint $H$ then
        $\mathcal{H} \leftarrow \mathcal{H} \cup H$                              ▷ Row generation
        continue
    else
        break                                                ▷ Found optimal solution!
```

situation of an airspace made of 400 sectors where 20 airports are placed randomly and a certain number of flights is scheduled (with random departure times) between two randomly chosen airports (see Figure 2). We must say that real-life data is available, but we are not allowed (yet) to use it. However, exploiting the information obtained from the real-life data, we selected 30 "realistic" random instances[2].

The algorithm has been implemented in C$^\sharp$ and interfaced with CPLEX 12.8 using its default parameters except for the following: the number of available threads was set to 1, the advanced start switch was set to 0, and both dual reduction and dynamic search were disabled. The information used for the advanced start (sometimes also called warm start) is poorly exploited by CPLEX in our context, and led to inconsistent results. Instead, dual reduction and dynamic search are automatically disabled by CPLEX when row generation is implemented using callback functions.

The results of Table 1 show a consistent and sometimes dramatic improvement in the solution time of the PC formulation. The strength of this new formulation (compared to the BF formulation) is demonstrated particularly by the smaller number of branch and bound nodes visited before reaching the optimal solution. This is mostly due to the absence of the large coefficient M in PC. In fact, the LP relaxation of BF at a particular node of the search tree is usually very poor, preventing an effective pruning of the branches of the search tree.

Overall, the *Path&Cycle* formulation for the Hotspot Problem proved to be very promising when compared to a more common formulation. Moreover, this formulation can be easily extended/modified to handle other job-shop scheduling problems.

References

1 Cyril Allignol, Nicolas Barnier, Pierre Flener, and Justin Pearson. Constraint programming for air traffic management: a survey 1: In memory of pascal brisset. *The Knowledge Engineering Review*, 27(3):361–392, 2012.

2 Thomas Dubot, Judicaël Bedouet, and Stéphane Degrémont. Modelling, generating and evaluating sector configuration plans-methodology report of the sesar vp-755 exercise. In *30th Congress of the International Council of the Aeronautical Sciences (ICAS 2016)*, 2016.

3 Leonardo Lamorgese and Carlo Mannino. A non-compact formulation for job-shop scheduling problems in transportation (Dagstuhl Seminar 16171). *Dagstuhl Reports*, 6(4):151, 2016. submited to Operations Research, under revision. URL: `http://drops.dagstuhl.de/opus/volltexte/2016/6694`, `doi:10.4230/DagRep.6.4.139`.

4 Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, 2002.

5 Tolebi Sailauov and ZW Zhong. An optimization model for large scale airspace. *International Journal of Modeling and Optimization*, 6(2):86, 2016.

6 Nina Schefers, Miquel Angel Piera, Juan José Ramos, and Jenaro Nosedal. Causal analysis of airline trajectory preferences to improve airspace capacity. *Procedia Computer Science*, 104:321–328, 2017.

7 Bo Vaaben and Jesper Larsen. Mitigation of airspace congestion impact on airline networks. *Journal of Air Transport Management*, 47:54–65, 2015.

[2] The instances are available from the authors upon request.

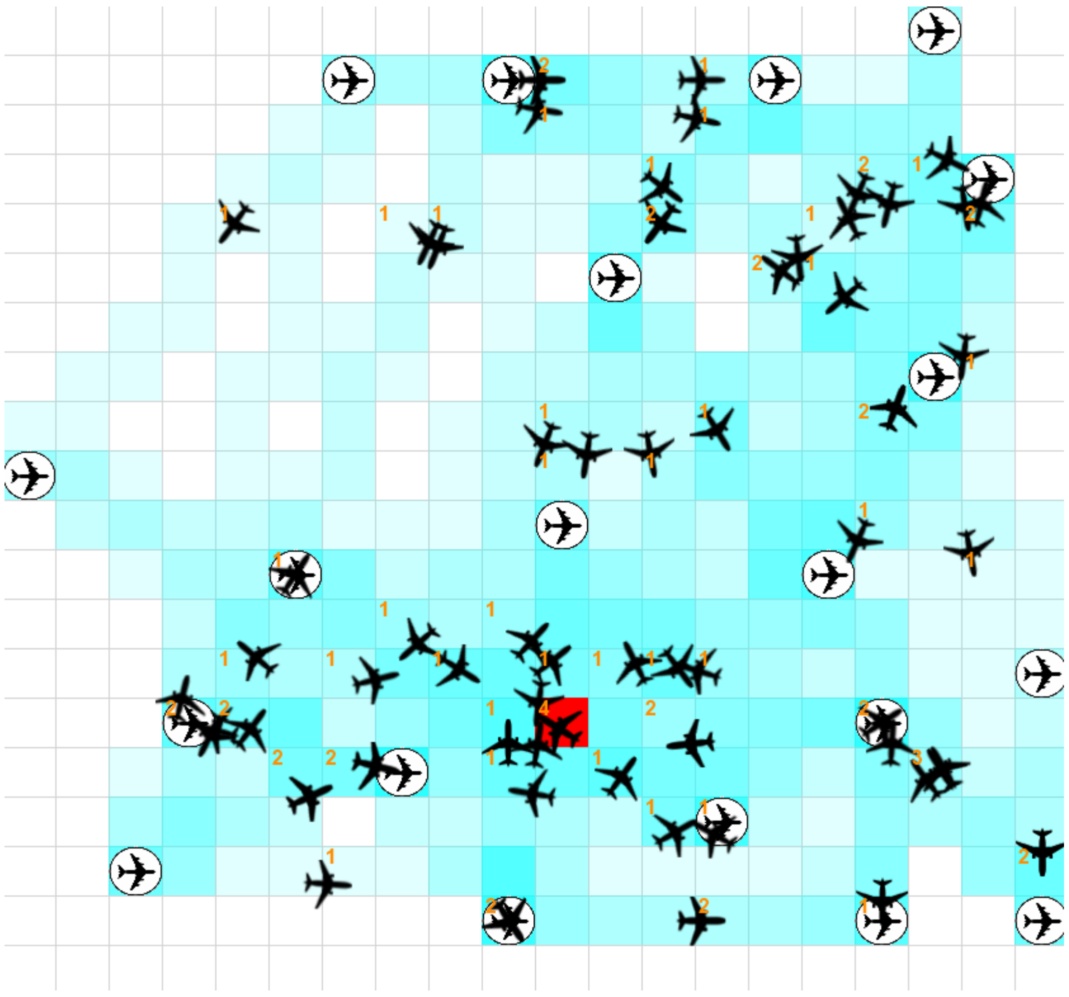

Figure 2 A snapshot of one of the instances where the sector capacity c_s is equal to 3, and a hotspot is highlighted in red. The orange number at the top left corner of each sector shows the current occupancy of the sector. The shades of turquoise simply indicate the number of flight routes traversing a particular sector, helping the visual analysis of an instance.

ID	$\|F\|$	c_s	Solved hotspots		Visited nodes		Time (s)		Speed up
			PC	BF	PC	BF	PC	BF	
ATM1	122	3	13	13	1016	19175	1.06	4.99	4.7x
ATM2	137	3	13	13	3062	36806	1.35	10.38	7.7x
ATM3	131	3	8	8	109	774	0.18	0.28	1.5x
ATM4	142	3	13	13	833	40482	0.73	6.43	8.8x
ATM5	110	3	12	12	795	31117	0.39	7.38	18.8x
ATM6	127	3	5	5	79	570	0.16	0.17	1.1x
ATM7	115	3	1	1	0	5	0.05	0.05	1.0x
ATM8	120	3	4	4	2	97	0.05	0.10	1.8x
ATM9	131	3	4	4	42	554	0.08	0.16	2.1x
ATM10	143	3	8	8	76	2313	0.18	0.48	2.6x
ATM11	136	3	15	15	371	39300	0.31	14.90	47.3x
ATM12	142	3	9	9	274	1974	0.22	0.57	2.6x
ATM13	139	3	11	11	118	2217	0.19	0.94	5.1x
ATM14	126	3	7	7	60	2182	0.13	0.59	4.6x
ATM15	139	3	9	9	3625	172950	0.88	50.61	57.4x
ATM16	288	5	4	4	47	1579	0.27	0.71	2.6x
ATM17	289	5	9	9	113	12503	0.38	6.05	16.0x
ATM18	278	5	6	6	183	2188	0.37	1.23	3.3x
ATM19	259	5	3	3	0	296	0.15	0.20	1.4x
ATM20	254	5	5	5	55	1977	0.23	1.01	4.3x
ATM21	279	5	6	6	255	4175	0.32	2.10	6.6x
ATM22	287	5	3	3	0	985	0.11	0.32	2.8x
ATM23	259	5	6	6	37	2452	0.25	1.00	4.0x
ATM24	281	5	4	4	161	1350	0.47	0.60	1.3x
ATM25	296	5	4	4	71	1518	0.22	0.60	2.7x
ATM26	275	5	7	7	50	2872	0.41	1.32	3.2x
ATM27	256	5	5	5	464	5042	0.46	1.58	3.5x
ATM28	273	5	6	6	298	1542	0.60	0.91	1.5x
ATM29	274	5	6	6	193	104627	0.53	35.09	66.7x
ATM30	287	5	7	7	1306	9129	0.75	3.10	4.1x

Table 1 Results on 30 randomly generated instances. The table shows the number of scheduled *flights*, the capacity c_s of the sectors (all the sectors have the same capacity), and how many *hotspots* have been solved by each algorithm (curiously, for these instances they are all equal, although this is not always the case since the hotspot constraints are added dynamically and each algorithm may take a different path to reach the optimal solution). It also reports the total number of *nodes* visited by the branch and bound algorithm, and the total *time* required to find the optimal solution. The last column is the ratio between the computation time of BF and the computation time of PC, and indicates the speed up obtained by using PC instead of BF.