

# Hotspot Resolution with Sliding Window Capacity Constraints using the Path&Cycle Algorithm

Carlo Mannino  
SINTEF and

Department of Mathematics,  
University of Oslo  
Oslo, Norway  
carlo.mannino@sintef.no

Andreas Nakkerud  
Department of Mathematics,  
University of Oslo  
Oslo, Norway  
andreana@math.uio.no

Giorgio Sartor  
SINTEF  
Oslo, Norway  
giorgio.sartor@sintef.no

Patrick Schittekat  
SINTEF  
Oslo, Norway  
patrick.schittekat@sintef.no

**Abstract**—We extend the new, efficient Path&Cycle formulation for the Hotspot Problem with two methods for dealing with windowed capacity constraints. We also discuss how to combine constraints to allow two-level capacity restrictions for peak and average load respectively. Finally, we present computational results for the sliding window capacity constraint.

## I. INTRODUCTION

The Hotspot Problem in Air Traffic Management is the problem of avoiding localized congestion in the controlled airspace. The airspace is divided into sectors, each with a capacity constraint. These constraints can limit the number of flights simultaneously in a region, or the number of flights entering a region in given time windows (see Figure 1). A *hotspot* [1], [2] is a sector with a violated capacity constraint. One common approach to eliminating hotspots is to limit the number of flights in fixed time windows. However, this will often result in *bunching* [3], where some of the flights in one window are moved to the beginning of the next, causing the beginnings of all windows to be crowded. In order to combat bunching, we propose to use sliding capacity windows.

In practice, it is not enough to only look at short-term peak capacity. Even when peak capacity is not violated, sustained high load puts too much strain on controllers. We propose combining capacity constraints to allow for both higher-load short-time peak capacity constraints and lower-load long-time capacity constraints. The ability to combine both short-term and long-term of capacity constraints in one model makes for more realistic hotspot resolution.

When checking for capacity violations, we count the number of flights currently in each sector, i.e., occupancy count. When using windowed constraints, another possibility is to count only the entries into the sector during the given window, i.e., entry count. The entry can be a better option if the workload associated with each flight is largely independent of the time that flight spends in the sector. In this paper, we only use occupancy counts, but our algorithms support the use of both counts interchangeably.

The Path&Cycle formulation is a new, efficient formulation for job-shop scheduling problems that was introduced in [4] to tackle the Hotspot Problem in Air Traffic Management. It does not have the disadvantages of time-indexed formulations

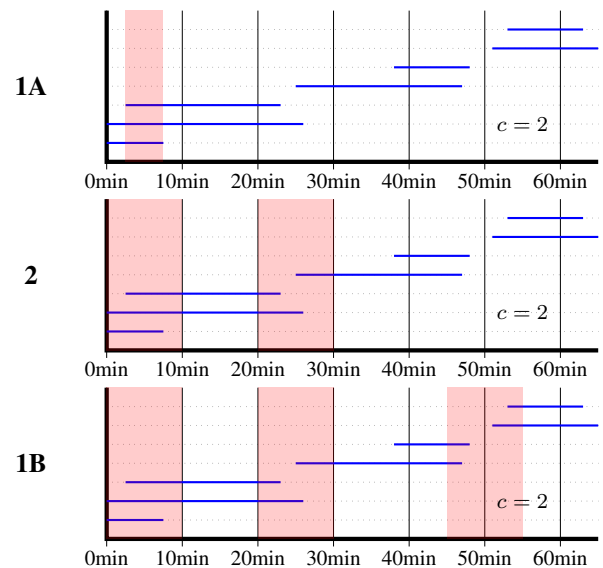


Figure 1. Three types of capacity constraints, each with capacity at most 2. Solid, blue lines represent flights present in this sector. Violations are shown in red. Type 1A violations are instantaneous, Type 2 violations are in fixed windows at 10 minute intervals, Type 1B violations are in any 10 minute interval.

and big-M formulations, which are used in similar approaches. In particular, time-indexed formulations struggle when the number of time periods grow large, while big-M formulations are slowed down due to weak bounds on optimality.

We briefly discuss the Path&Cycle model from [4] in Section II. We also introduce a new generalization of the Path&Cycle model which allows for sliding capacity windows. We discuss how to add a layer of fixed-window capacity constraints to the Path&Cycle model in Section III. Finally, we present computational results for sliding capacity windows in Section IV.

## II. THE PATH&CYCLE MODEL FOR TYPE 1 CAPACITY CONSTRAINTS

We are solving the Hotspot Problem for a set of sectors  $S$  and a set of flights  $F$ , under a variety of capacity constraints. A *route node*  $(f, s)$  is a pair of a flight and a sector, where

the flight  $f$  passes through the sector  $s$ . Each flight  $f \in F$  has an associated route, which is an ordered sequence of route nodes  $((f, s_1), (f, s_2), \dots, (f, s_q))$ , where  $s_1$  is the departure sector and  $s_q$  is the arrival sector. We let  $(f, s+1)$  denote the route node immediately following  $(f, s)$ . Our goal is to find a schedule  $t$  that specifies for each route node  $(f, s)$  the time  $t_f^s$  when flight  $f$  will enter sector  $s$ .

For each route node  $(f, s)$ , we are also given the time  $\Lambda_f^s$  flight  $f$  takes to traverse sector  $s$ . For each departure node, we are given the earliest departure time, and for some departure nodes we are also given the latest departure time. In addition to satisfying these time constraints, our schedule must also satisfy the given capacity constraints.

### A. Type 1 Capacity Constraints

Type 1 capacity constraints are constraints that apply to any interval (or instant) of a given length. Type 1A capacity constraints are violated if there is an instant where the capacity of a sector is exceeded. Type 1B capacity constraints are violated if there is an interval where the number of flights in the sector during that interval is above the capacity of the sector. (See Figure 1.)

The algorithm for Type 1A capacity violations (see Figure 1) from [4] is presented in Section II-B. Type 1B is a generalization of Type 1A, where each flight's occupancy in the sector is extended by the length of the desired capacity constraint window (see Figure 2 and Lemma 1). Since Type 1A is Type 1B with zero-width windows, we refer to the more general Type 1B as Type 1.

**Lemma 1.** *We have a Type 1B violation of capacity  $c$  and window width  $\Delta$  if and only if we have a Type 1A violation of capacity  $c$  where each flight occupancy has been extended by  $\Delta$  after the original occupancy.*

*Proof:* Suppose we have a Type 1B violation of capacity  $c$  and window width  $\Delta$ . Let  $t$  be the time at the end of the window. Now extend the duration of each flight by  $\Delta$ . Each flight present in the window will now either be present at  $t$ , or have its extension present at  $t$ , so we have a Type 1A violation at time  $t$ .

Suppose, for the other direction, that we have extended the durations of all flights by  $\Delta$ , and that we have a Type 1A violation of capacity  $c$  at time  $t$ . If we remove each extension, then all the flights with an extension present at  $t$  must have been present between  $t - \Delta$  and  $t$ , so we have a Type 1B violation of capacity  $c$  and window width  $\Delta$ . ■

### B. The Path&Cycle Formulation for Type 1A

We first develop the algorithms and models needed to solve for Type 1A capacity constraints. We then describe the generalization to Type 1B in Section II-D.

Let  $t_f^s$  be the time flight  $f$  enters sector  $s$ , and let  $\Lambda_f^s$  be the time flight  $f$  takes to traverse sector  $s$ . When the flight is understood from context, we use  $s+1$  to denote the next sector in the flight's path. We get the equality

$$t_f^{s+1} = t_f^s + \Lambda_f^s, \quad (1)$$

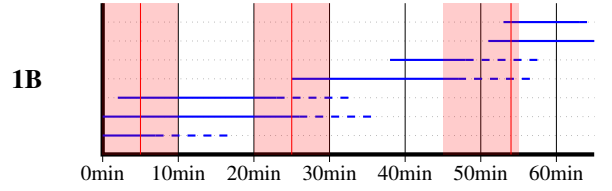


Figure 2. By extending the duration of each flight's occupation of the region, we can turn 1B violations into 1A violations. We use dashed lines to represent occupancy extensions.

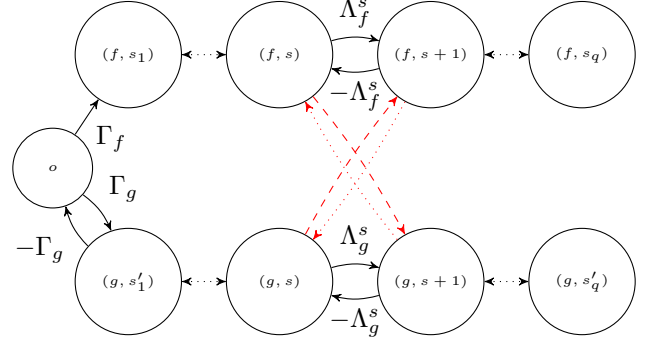


Figure 3. Route node graph. The black edges between route nodes of a single flight represent departure constraints and sector traversal times. The red edges between route nodes on different flights (conflict edges) represent scheduling constraints between flights. In this example  $f$  has an earliest departure time, while  $g$  has a fixed departure time.

which we will represent using the inequalities  $t_f^{s+1} \geq t_f^s + \Lambda_f^s$  and  $t_f^s \geq t_f^{s+1} - \Lambda_f^s$ .

The earliest departure time, relative to the reference time  $t_o$ , of a flight  $f$  is denoted  $\Gamma_f$ . Thus

$$t_f^{s_1} \geq t_o + \Gamma_f. \quad (2)$$

If the flight must depart on schedule, then we also add the reverse inequality to make an equality. In our problem, we assume flights cannot be delayed in the air. Therefore, flights arriving from outside the managed area, and flights already in the air, are accounted for using fixed departure times.

We represent (1) and (2) using a weighted, directed graph where the nodes are route nodes, and the weighted, directed edges represent inequalities. Figure 3 shows the resulting graph.

When there is an edge from  $(f, s)$  to  $(f', s')$ , with weight  $w$ , this means that  $t_{f'}^{s'} \geq t_f^s + w$ . The red, diagonal edges between route nodes of different flights in Figure 3 represent possible *conflict edges*. The dashed edges (going left to right) together represent a meeting of  $f$  and  $g$  in  $s$ . The two constraints together require that both flights enter  $s$  before either leaves. The dotted edges (right to left) each represent a precedence constraint. In each case, one flight has to leave  $s$  before the other enters. See [5] for more details on disjunctive graphs.

Associated to the conflict edges, we introduce the variables

$$x_{fg}^s = \begin{cases} 1 & \text{if } f \text{ and } g \text{ meet in } s, \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

$$y_{fg}^s = \begin{cases} 1 & \text{if } f \text{ precedes } g \text{ in } s, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Note that for any pair of flights  $f, g$ , and for any region  $s$ , we must have

$$y_{fg}^s + y_{gf}^s + x_{fg}^s = 1. \quad (5)$$

We let  $G(\mathbf{y}, \mathbf{x})$  be the route node graph with added conflict edges such that  $\mathbf{x}$  and  $\mathbf{y}$  are incidence matrices. We divide the set  $K$  of all conflict edges into the two sets  $K_x$  and  $K_y$ . Note that  $K$ ,  $K_x$ , and  $K_y$  contains all conflict edges of the matching type, not only those currently selected.

If  $y_{fg}^s = 1$ , we add the inequality

$$t_f^{s+1} \leq t_g^s, \quad (6)$$

that is,  $f$  leaves  $s$  before  $g$  enters. If  $x_{fg}^s = 1$ , we add the pair of inequalities

$$t_f^{s+1} \geq t_g^s \quad (7)$$

$$t_g^{s+1} \geq t_f^s, \quad (8)$$

that is, neither may leave  $s$  before both have arrived.

**Lemma 2.** *If  $G(\mathbf{y}, \mathbf{x})$  contains a strictly positive directed cycle, then the set of inequalities corresponding to the edges of the cycle is inconsistent (infeasible).*

*Proof:* If  $G(\mathbf{y}, \mathbf{x})$  contains a directed cycle visiting node  $(f, s)$ , and the sum total weight of the edges in the cycle is  $W > 0$ , then

$$t_f^s \geq t_f^s + W > t_f^s. \quad \blacksquare$$

From Lemma 2, it follows that  $G(\mathbf{y}, \mathbf{x})$  cannot contain strictly positive directed cycles. This restricts the possible values of  $\mathbf{x}$  and  $\mathbf{y}$ .

Given a  $G$  with no strictly positive directed cycles, we can always find a longest path from  $o$  to any route node  $u = (f, s)$ . We label this distance  $L^*(\mathbf{y}, \mathbf{x}, u)$ .

**Lemma 3.** *Let  $\mathbf{x}$  and  $\mathbf{y}$  be such that  $G(\mathbf{y}, \mathbf{x})$  does not contain any strictly positive directed cycles, let  $f$  be a flight, and  $s$  be a sector. Minimizing  $t_f^s$  subject to the instances of (6), (7), and (8) corresponding to conflict edges in  $G(\mathbf{y}, \mathbf{x})$  is equivalent to minimizing  $L^*(\mathbf{y}, \mathbf{x}, (f, s))$ .*

*Proof:* By the same argument as in the proof of Lemma 2, we have that

$$t_u \geq t_o + L^*(\mathbf{y}, \mathbf{x}, u).$$

This inequality is most restricting, since  $L^*$  is the longest path. Therefore, when minimizing  $t_f^s$ , we can instead minimize  $L^*(\mathbf{y}, \mathbf{x}, (f, s))$ .  $\blacksquare$

Lemma 3 is the key to building a model with no direct reference to the scheduling variables. All connections to time are coded into the edge weights of the route node graph.

### C. The Mixed-Integer Linear Programming Model

The variables for our model are  $\mathbf{x}$  and  $\mathbf{y}$ , introduced in Section II-B. For each sector, these variables encode meetings of flights, and precedence between flights that do not meet.

Lemmas 2 and 3 give us most of what we need to build our model. All the scheduling and conflict inequalities are encoded in the graph  $G(\mathbf{y}, \mathbf{x})$ . The only thing missing is the encoding of capacity constraints.

**Lemma 4.** *Let  $F$  be the set of all flights,  $s$  a sector, and  $c_s$  the capacity of  $s$ . The capacity constraint  $c_s$  is at all times respected if and only if for all  $\bar{F} \subseteq F$  where  $|\bar{F}| = c_s + 1$ , we have*

$$\sum_{\{f,g\} \subseteq \bar{F}} x_{fg}^s \leq \binom{|\bar{F}|}{2} - 1. \quad (9)$$

*Proof:* Suppose the capacity is violated at some point in time. At that time, at least  $c_s + 1$  flights must be in  $s$ . Let  $\bar{F}$  contain any  $c_s + 1$  of these flights. The number of pairs in  $\bar{F}$  is  $\binom{|\bar{F}|}{2}$ . Since each pair is meeting, the sum in (9) is  $\binom{|\bar{F}|}{2}$ , and so the inequality is violated.

Conversely, suppose (9) is violated. Then there is a set  $\bar{F}$  of  $c_s + 1$  flights with at least  $\binom{|\bar{F}|}{2}$  pairwise meetings. Since this is the total number of possible meetings, the flights must all meet. Since they all meet, none may leave  $s$  before all have entered, and so there is a point in time when they are all present, and the capacity is violated.  $\blacksquare$

**Lemma 5.** *Let  $\mathcal{C}$  be the set of all strictly positive directed cycles in  $G(\mathbf{1}, \mathbf{1})$ .  $G(\mathbf{y}, \mathbf{x})$  contains the strictly positive directed cycle  $C \in \mathcal{C}$  if and only if*

$$\sum_{e \in C \cap K_y} y_e + \sum_{e \in C \cap K_x} x_e = |C \cap K|.$$

*Proof:*  $C$  is a subgraph of  $G(\mathbf{y}, \mathbf{x})$  exactly when all conflict edges in  $C$ , numbering  $|C \cap K|$ , are selected.  $\blacksquare$

One typical objective is to minimize the sum of delays of all flights. This is equivalent to minimizing the sum of arrival times, since our scheduling constraints make it impossible to schedule early arrivals. If we let  $A$  be the set of arrival nodes, then our goal is to minimize  $\sum_{u \in A} t_u$ . Using the results of Section II-B and Lemmas 4 and 5, we get the following Linear Programming model.

$$\begin{aligned} \min \quad & \sum_{u \in A} L^*(\mathbf{y}, \mathbf{x}, u) \\ \text{s.t.} \quad & (i) \quad y_{fg}^s + y_{gf}^s + x_{fg}^s = 1, \quad \{f, g\} \subseteq F, s \in S, \\ & (ii) \quad \sum_{e \in C \cap K_y} y_e + \sum_{e \in C \cap K_x} x_e \leq |C \cap K| - 1, \\ & \quad \quad \quad C \in \mathcal{C}, \\ & (iii) \quad \sum_{\{f,g\} \subseteq \bar{F}} x_{fg}^s \leq \binom{|\bar{F}|}{2} - 1, \\ & \quad \quad \quad s \in S, \bar{F} \subseteq F, |\bar{F}| = c_s + 1, \\ & \mathbf{y} \in \{0, 1\}^{|K_y|}, \mathbf{x} \in \{0, 1\}^{|K_x|}. \end{aligned} \quad (10)$$

#### D. Generalization to Type 1B

By Lemma 1 (see also Figure 2), we know that we model Type 1B constraints by modifying (6), (7), and (8). We use  $\Delta$  to denote the width of our Type 1B capacity windows. For  $y_{fg}^s = 1$ , we get

$$t_f^{s+1} + \Delta \leq t_g^s, \quad (11)$$

and for  $x_{fg}^s$  we get

$$t_f^{s+1} + \Delta \geq t_g^s \quad (12)$$

$$t_g^{s+1} + \Delta \geq t_f^s. \quad (13)$$

The only required change is modification of weights in the route node graph  $G$ .

#### E. Row and Column Generation

Model (10) is well suited to delay row and column generation. First, we need only generate  $y_{fg}^s, y_{gf}^s, x_{fg}^s$  and the associated row of type (10.i) if  $f$  and  $g$  are violating a capacity constraint.

Of the constraints of type (10.ii) and (10.iii), most rows will not be relevant. Further, in the relevant rows, we do not need to worry about ungenerated variables, as we always generate the  $x$ 's that may take value 1, i.e., those that represent flights that may meet in the given sector.

Dealing with the objective function is the challenging aspect of delayed row and column generation in this model. This is because the longest path from  $o$  to any  $u \in A$  will depend on the choice of  $\mathbf{x}$  and  $\mathbf{y}$  through  $G(\mathbf{y}, \mathbf{x})$ .

Let  $\mathcal{H}$  be the set of all  $G(\mathbf{y}, \mathbf{x})$ , such that  $\mathbf{x}$  and  $\mathbf{y}$  satisfy (10.i), (10.ii), and (10.iii). We use  $P_u(H)$  to denote (the set of edges of) a longest path from  $o$  to  $u$  in  $H$  for  $u \in A$  and  $H \in \mathcal{H}$ .  $L_u(H)$  is the length of  $P_u(H)$ . If all the conflict edges in  $H$  are chosen by the current solution, then  $L_u(H) = L^*(\mathbf{y}, \mathbf{x}, u)$ .

If all the conflict edges of  $H$  are selected, then

$$\sum_{e \in P_u(H) \cap K_x} x_e + \sum_{e \in P_u(H) \cap K_y} y_e = |K \cap P_u(H)|. \quad (14)$$

That is, the set of inequalities

$$L_u(H) \left( \sum_{e \in P_u(H) \cap K_x} x_e + \sum_{e \in P_u(H) \cap K_y} y_e - |K \cap P_u(H)| + 1 \right) \leq \mu_u, \quad H \in \mathcal{H}, \quad (15)$$

is equivalent to

$$L^*(\mathbf{y}, \mathbf{x}, u) \leq \mu_u. \quad (16)$$

Thus, by expressing the objective of (10) in terms of  $\mu_u$  and adding the inequalities (15), we obtain the Path&Cycle formulation. When solving this model, we can start with  $\mathcal{H} = \mathcal{C} = \emptyset$ , and only add inequalities for longest paths ( $H$ ) and cycles ( $C$ ) when they become relevant. The steps of the delayed row and column generation algorithm are described in detail in [4].

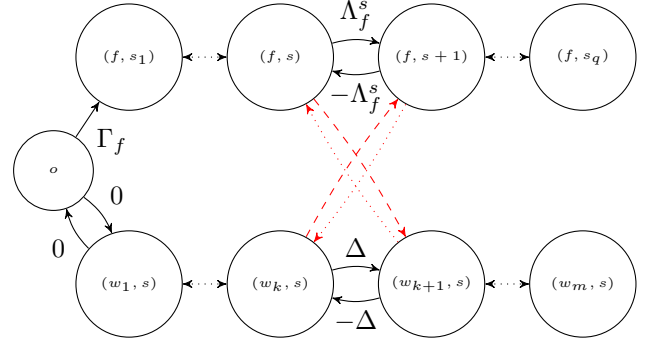


Figure 4. The figure shows how we can adapt the route node graph to model Type 2 capacity constraints. The lower line of nodes represent capacity windows for a single sector  $s$ , which is visited by flight  $f$ .

### III. MODELLING TYPE 2 CAPACITY WINDOWS

We now present a way to add Type 2 capacity constraints to our current model. This allows us to use two different, simultaneous capacity constraints.

We adapt existing methods in order to model Type 2 capacity constraints. Figure 4 shows how we modify the route node graph in order to account for capacity windows. We can make a further simplification by linking the window nodes directly to the origin  $o$  (see Figure 5).

For each sector  $s$  and each time window  $w$ , we introduce a new *window node*  $(w, s)$ . We extend  $\mathbf{x}$  and  $\mathbf{y}$  by thinking of each  $w$  as a flight. This means that

$$x_{fw}^s = \begin{cases} 1 & f \text{ is in } s \text{ in window } w, \\ 0 & \text{otherwise,} \end{cases} \quad (17)$$

$$y_{fw}^s = \begin{cases} 1 & f \text{ leaves } s \text{ before window } w \text{ begins,} \\ 0 & \text{otherwise,} \end{cases} \quad (18)$$

$$y_{wf}^s = \begin{cases} 1 & f \text{ enters } s \text{ after window } w \text{ ends,} \\ 0 & \text{otherwise.} \end{cases} \quad (19)$$

We use  $t_w$  to denote the start time of window  $w$ , the end time is then  $t_w + \Delta$ . We let  $m$  be the number of windows, and label the windows  $w_1, \dots, w_m$ . For each  $i \leq m$  and for each sector  $s$ , we add the window node  $(w_i, s)$  to the route node graph  $G$ . We also add, for each  $i \leq m$ , edges corresponding to the pair of inequalities

$$t_{w_i}^s = t_o + (i-1)\Delta. \quad (20)$$

That is, and edge from  $o$  to  $(w_i, s)$  with weight  $(i-1)\Delta$ , and an edge from  $(w_i, s)$  to  $o$  with weight  $-(i-1)\Delta$ .

If  $x_{fw}^s = 1$ , we add the inequalities (and corresponding edges)

$$t_f^s \leq t_w^s + \Delta \quad (21)$$

$$t_f^{s+1} \geq t_w^s, \quad (22)$$

if  $y_{fw}^s = 1$ , we add the inequality

$$t_f^{s+1} \leq t_w^s, \quad (23)$$

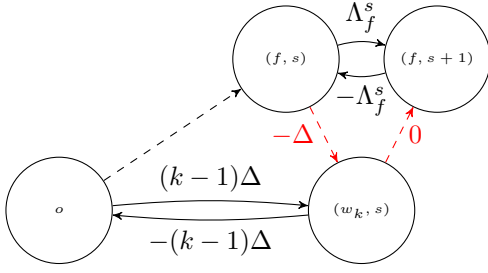


Figure 5. Sub graph of the route node graph. This subgraph encodes the fact that flight  $f$  is in sector  $s$  in window  $w_k$ .

and if  $y_{wf}^s = 1$ , we add

$$t_f^s \geq t_w^s + \Delta. \quad (24)$$

We are assuming, without loss of generality, a fixed window size  $\Delta$ . In order to use a variable window size  $\Delta_w^s$ , simply modify (20) as follows

$$t_{w_i}^s = t_o + \sum_{j=1}^{i-1} \Delta_{w_j}^s, \quad (25)$$

and replace  $\Delta$  with  $\Delta_w^s$  in (21) and (24). Note that  $\Delta_w^s$  is free to vary by window and sector, so that each sector can use its own set of windows.

Figure 5 shows the edges added to the route node graph when flight  $f$  is in region  $s$  in window  $w_k$ . The edges between  $o$  and  $(w_k, s)$  represent (20), the edges between the window node and the route nodes represent (21) and (22).

So far, we have extended the route node graph  $G(\mathbf{y}, \mathbf{x})$  in order to account for the scheduling constraints for capacity windows. What remains is to define proper constraints on the new variables in  $\mathbf{x}$  and  $\mathbf{y}$ . We let  $W$  be the set of windows. As before, we have

$$y_{fw}^s + y_{wf}^s + x_{fw}^s = 1, \quad f \in F, w \in W, s \in S. \quad (26)$$

The added capacity constraint is simpler, since we now only need to count the number of flights that appear in each window. We let  $c_s^w$  be the capacity for window  $w$  and sector  $s$ , then

$$\sum_{f \in F} x_{fw}^s \leq c_s^w, \quad s \in S, w \in W. \quad (27)$$

By adding these new inequalities to (10), we get the following model.

$$\begin{aligned} \min \quad & \sum_{u \in A} L^*(\mathbf{y}, \mathbf{x}, u) \\ \text{s.t.} \quad & \\ (i.a) \quad & y_{fg}^s + y_{gf}^s + x_{fg}^s = 1, \quad \{f, g\} \subseteq F, s \in S, \\ (i.b) \quad & y_{fw}^s + y_{wf}^s + x_{fw}^s = 1, \quad f \in F, w \in W, s \in S, \\ (ii) \quad & \sum_{e \in C \cap K_y} y_e + \sum_{e \in C \cap K_x} x_e \leq |C \cap K| - 1, \\ & C \in \mathcal{C}, \\ (iii.a) \quad & \sum_{\{f, g\} \subseteq \bar{F}} x_{fg}^s \leq \binom{|\bar{F}|}{2} - 1, \\ & s \in S, \bar{F} \subseteq F, |\bar{F}| = c_s + 1, \\ (iii.b) \quad & \sum_{f \in F} x_{fw}^s \leq c_s^w \quad s \in S, w \in W, \\ & \mathbf{y} \in \{0, 1\}^{|\mathcal{K}_y|}, \mathbf{x} \in \{0, 1\}^{|\mathcal{K}_x|}. \end{aligned} \quad (28)$$

This model is suited for the same delayed row and column generation as in Section II-E.

#### IV. COMPUTATIONAL RESULTS

Tables I and II show the results of our experiments with Type 1 capacity constraints, based on simulated data. The first columns of Table I shows the performance of our Type 1A algorithm. The running times are longer in the cases with Type 1B capacity windows of 10 seconds and 1 minute, but this is expected; there are more hotspots when we use time windows, as shown in Figure 1. The data in Table I also confirms that the algorithms have to resolve more hotspots for the wider windows.

In our experiments, the Path&Cycle formulation solves all test instances within a few seconds, while the standard Big-M formulation times out at 10 minutes on some of the more difficult instances, especially with the longer 1 minute windows.

In Table II, we show results from the same instances as in Table I with the same capacity (left-most columns), but using 10 minute Type 1B windows. In this case, the number of hotspots grew too large and almost all of the computations timed out at 10 minute limit. In the worst instances, over 100 hotspots were resolved before time ran out.

We have designed our instances to have a reasonable amount of hotspots with Type 1A capacity constraints. For a proper test of the Type 1B constraints, we need to increase the capacity or change the instances to reduce the number of hotspots. In Table II, we also show the effect of increasing the capacity of the sectors. As the capacity increases, the number of hotspots go down, and many more instances are solved within the time limit.

Our experiments were done with a C# implementation using CPLEX 12.8. CPLEX was set to default parameters, except the number of available threads were set to 1, the advanced start switch was set to 0, and both dual reduction and dynamic search were disabled. The code was run on an Intel i7-7700 HQ 2.8 GHz CPU, with 32 GB of RAM.

TABLE I

COMPUTATIONAL RESULTS FOR TYPE 1A AND TYPE 1B CAPACITY CONSTRAINTS. PATH&CYCLE RESULTS ARE LABELED PC, BIG-M FORMULATION RESULTS ARE LABELED BM. TIME LIMIT WAS SET TO 600 SECONDS, AND SOME BM COMPUTATIONS TIMED OUT. THE NUMBER OF HOTSPOTS INCREASES WITH THE WIDTH OF THE CAPACITY WINDOWS, THIS IN TURN INCREASES COMPUTATION TIME. THE "NODES" COLUMNS SHOW HOW MANY NODES WERE VISITED BY THE BRANCH AND BOUND ALGORITHM USED BY THE MILP SOLVER IN CPLEX.

F	$c_s$	Type 1A						Type 1B, 10 seconds time window						Type 1B, 1 minute time window					
		Hotspots		Nodes		Time (s)		Hotspots		Nodes		Time (s)		Hotspots		Nodes		Time (s)	
		PC	BM	PC	BM	PC	BM	PC	BM	PC	BM	PC	BM	PC	BM	PC	BM	PC	BM
122	3	16	15	1e+3	9e+4	1.24	10.99	17	16	2e+3	9e+5	1.45	168.4	20	20	2e+4	3e+6	4.16	449.3
137	3	21	21	5e+3	2e+6	2.24	117.9	21	21	7e+3	1e+7	2.17	494.3	19	15	7e+3	3e+6	2.22	—
131	3	12	12	3e+2	3e+4	0.48	2.99	12	12	3e+2	5e+4	0.49	4.79	13	13	1e+3	3e+4	0.62	2.97
142	3	13	13	5e+2	1e+5	0.53	29.93	13	13	9e+2	1e+5	0.8	26.34	17	17	3e+3	8e+5	1.99	118.5
110	3	12	12	5e+2	4e+4	0.29	9.82	12	12	5e+2	5e+4	0.4	11.55	16	16	9e+3	8e+5	1.58	113.7
127	3	11	11	2e+2	1e+4	0.35	1.04	12	12	3e+2	9e+3	0.52	0.89	18	18	2e+3	2e+4	1.09	1.87
115	3	1	1	0e+0	7e+0	0.05	0.04	1	1	0e+0	6e+0	0.04	0.04	1	1	0e+0	7e+0	0.04	0.04
120	3	4	4	8e+0	1e+1	0.04	0.07	4	4	3e+0	5e+1	0.04	0.06	5	5	9e+0	5e+1	0.04	0.07
131	3	7	7	3e+1	2e+3	0.12	0.36	7	7	3e+1	2e+3	0.13	0.34	8	8	5e+1	5e+3	0.12	0.78
143	3	8	8	6e+1	2e+3	0.14	0.46	8	8	2e+1	2e+3	0.14	0.75	10	10	2e+2	3e+4	0.2	6.42
136	3	15	15	5e+2	6e+4	0.29	17.21	17	17	4e+2	2e+6	0.28	—	20	20	2e+3	2e+6	0.63	—
142	3	9	9	2e+2	6e+3	0.1	1.68	9	9	3e+2	5e+3	0.16	1.41	12	12	1e+3	1e+5	0.45	14.18
139	3	14	14	1e+2	2e+4	0.27	4.42	14	14	1e+2	7e+4	0.29	8.83	20	20	8e+2	6e+5	0.73	76.77
126	3	10	10	2e+2	7e+3	0.24	1.38	11	11	4e+2	1e+4	0.33	1.9	15	14	2e+3	3e+5	0.77	49.75
139	3	19	19	1e+4	2e+5	2.16	31.16	19	19	1e+4	1e+6	1.99	266.7	24	24	3e+4	1e+6	5.29	134.9
288	5	8	8	8e+1	7e+3	0.54	1.52	8	8	7e+1	5e+3	0.51	1.24	12	12	6e+2	1e+5	1.32	18.41
289	5	9	9	3e+1	2e+4	0.23	7.31	10	10	7e+1	2e+4	0.26	9.41	14	14	3e+3	2e+6	1.95	—
278	5	10	10	4e+2	4e+4	0.92	9.88	11	11	5e+2	5e+4	1	12.29	16	14	7e+3	2e+6	3.74	—
259	5	3	3	0e+0	5e+2	0.1	0.23	3	3	0e+0	4e+2	0.11	0.2	6	6	2e+1	8e+2	0.15	0.44
254	5	8	8	7e+1	7e+3	0.31	2.09	8	8	7e+1	8e+3	0.41	2.1	9	9	6e+2	1e+5	0.58	22.85
279	5	9	9	5e+2	4e+4	0.66	8.37	9	9	8e+2	1e+4	0.82	3.91	14	14	5e+3	2e+6	2.38	—
287	5	3	3	0e+0	4e+2	0.12	0.24	6	6	0e+0	2e+3	0.33	0.57	10	10	6e+1	4e+3	0.36	1.78
259	5	11	11	8e+1	1e+4	0.59	2.52	14	14	1e+2	4e+4	0.86	8.96	16	16	7e+2	7e+5	1.63	136.3
281	5	8	8	2e+2	9e+3	0.75	1.85	9	9	2e+2	2e+4	0.83	3.83	12	12	5e+2	8e+4	1.28	23.86
296	5	4	4	4e+1	1e+3	0.16	0.67	4	4	4e+1	2e+3	0.16	0.93	6	6	2e+2	8e+3	0.5	1.8
275	5	7	7	2e+1	8e+2	0.24	0.69	7	7	2e+1	3e+3	0.25	1.46	8	8	2e+2	5e+4	0.39	16.78
256	5	5	5	2e+2	3e+3	0.37	0.83	5	5	2e+2	3e+3	0.4	0.78	7	7	7e+2	3e+3	0.7	1
273	5	9	9	2e+2	1e+4	0.6	3.14	9	9	3e+2	4e+4	0.69	12.37	12	12	2e+3	2e+5	1.92	32.34
274	5	9	9	7e+1	2e+4	0.67	4	9	9	5e+2	2e+4	0.84	4.77	16	16	5e+3	3e+6	3.37	—
287	5	11	11	1e+3	2e+4	0.89	5.34	12	12	1e+3	4e+4	1.48	9.93	16	16	5e+3	1e+6	4.18	193.5

## V. CONCLUSIONS

We have shown (see Section IV) that our algorithm for Type 1A (see [4]) also efficiently solves for Type 1B capacity constraints with short time windows, and that it performs well compared to the standard Big-M formulation on almost all our instances.

With a large number of flights, low capacities, and long capacity windows, the number of hotspots becomes too large to handle. However, according to the results in Tables I and II, the Path&Cycle formulation can easily handle up to 20–30 hotspots (including those introduced by intermediate solutions).

In order to further prove our model, we need access to real instances. While we have shown that our model performs well compared to one established approach, we still need to confirm that it performs well in real life.

## VI. FUTURE WORK

Our next step is to integrate Type 2 constraints into our model, so that we can solve the Hotspot Problem with layered

capacity constraints. This will allow us to restrict, simultaneously, peak and average load. By tuning window sizes, it is possible to approximate a wide range of capacity constraint schemes. We will also introduce entry counts as an alternative to occupancy counts for defining capacity constraints.

Using this model in practice would allow for more realistic modeling of the air traffic controller's workload capacity constraints, and therefore result in a more achievable work load. Also, the use of sliding windows have the added benefit of reducing the drive towards bunching.

Furthermore, our experiments have indicated that the Path&Cycle algorithm is well suited to reoptimization with slight variations in the model. This makes the algorithm ideal for the approach to inter-airline scheduling fairness presented by Jacquillat and Vaze [6]. We will include this approach to fairness in our future models, so that we can evaluate the fairness criteria themselves in terms of performance and effectiveness.

TABLE II

COMPUTATIONAL RESULTS FOR TYPE 1B CAPACITY CONSTRAINTS WITH 10 MINUTE WINDOWS. PATH&CYCLE RESULTS ARE LABELED PC, BIG-M FORMULATION RESULTS ARE LABELED BM. TIME LIMIT WAS SET TO 600 SECONDS. AT THE LOWER CAPACITIES, THE NUMBER OF HOTSPOTS GROW VERY LARGE, AND ALMOST ALL COMPUTATIONS TIME OUT. AS THE CAPACITY INCREASES, THE NUMBER OF HOTSPOTS BECOMES MANAGEABLE.

F	Type 1B, 10 minutes time window					Type 1B, 10 minutes time window					Type 1B, 10 minutes time window				
	$c_s$	Hotspots		Time (s)		$c_s$	Hotspots		Time (s)		$c_s$	Hotspots		Time (s)	
		PC	BM	PC	BM		PC	BM	PC	BM		PC	BM	PC	BM
122	3	61	47	—	—	4	11	11	1.49	61.2	5	1	1	0.19	0.07
137	3	75	58	—	—	4	16	16	1.68	47.78	5	1	1	0.07	0.06
131	3	52	47	—	—	4	5	5	0.1	0.18	5	0	0	0.03	0.04
142	3	52	48	—	—	4	8	8	0.13	0.28	5	0	0	0.02	0.03
110	3	47	47	—	—	4	4	4	0.06	0.32	5	0	0	0.02	0.02
127	3	58	63	—	—	4	19	19	4.57	74.73	5	2	2	0.06	0.08
115	3	14	14	0.2	53.24	4	0	0	0.02	0.03	5	0	0	0.02	0.03
120	3	33	32	43.76	—	4	1	1	0.05	0.04	5	0	0	0.03	0.02
131	3	50	35	—	—	4	6	6	0.11	0.47	5	0	0	0.03	0.04
143	3	51	54	—	—	4	2	2	0.04	0.07	5	0	0	0.03	0.03
136	3	65	62	—	—	4	7	7	0.36	8.25	5	0	0	0.02	0.03
142	3	54	53	—	—	4	9	9	0.79	24.43	5	0	0	0.03	0.03
139	3	63	50	—	—	4	9	9	0.58	2.68	5	1	1	0.05	0.06
126	3	68	71	—	—	4	16	16	1.22	5.46	5	2	2	0.05	0.08
139	3	76	56	—	—	4	21	21	8.19	—	5	3	3	0.04	0.06
288	5	59	60	—	—	6	31	24	39.06	—	7	2	2	0.18	0.26
289	5	102	103	—	—	6	46	36	—	—	7	7	7	0.29	1.32
278	5	0	142	—	—	6	48	41	—	—	7	4	4	0.2	0.52
259	5	57	58	—	—	6	37	25	—	—	7	7	7	0.49	5.05
254	5	44	44	—	—	6	25	14	53.25	—	7	6	6	0.79	0.82
279	5	75	76	—	—	6	29	25	—	—	7	9	9	0.47	1.47
287	5	84	83	—	—	6	42	27	—	—	7	4	4	0.22	0.7
259	5	87	86	—	—	6	27	20	487.6	—	7	2	2	0.14	0.21
281	5	91	89	—	—	6	43	31	—	—	7	11	11	2.57	5.07
296	5	60	62	—	—	6	24	21	213.2	—	7	2	2	0.12	0.24
275	5	68	67	—	—	6	16	16	18.31	—	7	1	1	0.11	0.13
256	5	56	57	—	—	6	32	19	—	—	7	6	6	0.39	1.14
273	5	0	97	—	—	6	48	29	—	—	7	10	10	2.72	8.05
274	5	93	93	—	—	6	44	37	—	—	7	10	10	1.41	—
287	5	0	112	—	—	6	36	24	—	—	7	8	8	1.24	14.26

## REFERENCES

- [1] C. Allignol, N. Barnier, P. Flener, and J. Pearson, "Constraint programming for air traffic management: a survey: In memory of pascal brisset," *The Knowledge Engineering Review*, vol. 27, no. 3, p. 361392, 2012.
- [2] T. Dubot, J. Bedouet, and S. Degrumont, "Modelling, generating and evaluating sector configuration plans," in *30th Congress of the International Council of the Aeronautical Sciences (ICAS 2016)*, 2016.
- [3] S. Stoltz and P. Ky, "Reducing traffic bunching through a more flexible air traffic flow management," in *4th USA/R&D Seminar*, 2001.
- [4] C. Mannino and G. Sartor, "The Path&Cycle Formulation for the Hotspot Problem in Air Traffic Management," in *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*, ser. OpenAccess Series in Informatics (OASICs), R. Borndörfer and S. Storandt, Eds., vol. 65. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 14:1–14:11. [Online]. Available: <http://drops.dagstuhl.de/opus/volltexte/2018/9719>
- [5] A. Mascis and D. Pacciarelli, "Job-shop scheduling with blocking and no-wait constraints," *European Journal of Operational Research*, vol. 143, no. 3, pp. 498 – 517, 2002. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0377221701003381>
- [6] A. Jacquillat and V. Vaze, "Interairline equity in airport scheduling interventions," *Transportation Science*, vol. 52, no. 4, 2018.