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 Δ to denote the width of our Type 1B capacity windows. For $y^s_{fg} = 1$, we get

$$t^{s+1}_f + \Delta \leq t^s_g, \tag{11}$$

and for x^s_{fg} we get

$$t^{s+1}_f + \Delta \geq t^s_g \tag{12}$$
$$t^{s+1}_g + \Delta \geq t^s_f. \tag{13}$$

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 $y^s_{fg}, y^s_{gf}, x^s_{fg}$ 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 off 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

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

That is, the set of inequalities

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

is equivalent to

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

Thus, by expressing the objective of (10) in terms of μ_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].

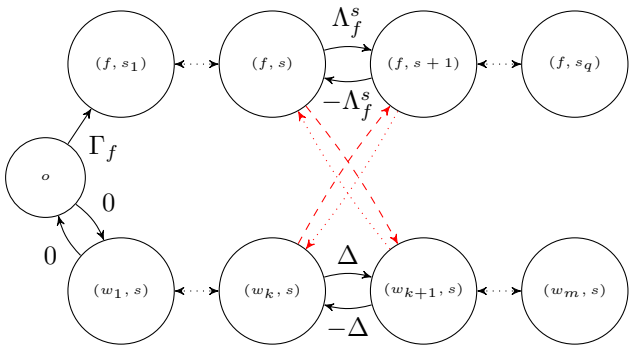

Figure 4. The figure shows how we can adapt the route node graph to model Type 2 capacity constraints. The lower line of nodes represent capacity windows for a single sector s, which is visited by flight f.

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

$$x^s_{fw} = \begin{cases} 1 & f \text{ is in } s \text{ in window } w, \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$y^s_{fw} = \begin{cases} 1 & f \text{ leaves } s \text{ before window } w \text{ begins,} \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

$$y^s_{wf} = \begin{cases} 1 & f \text{ enters } s \text{ after window } w \text{ ends,} \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

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 $w_1, \ldots, w_m$. 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

$$t^s_{w_i} = t_o + (i-1)\Delta. \tag{20}$$

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 $x^s_{fw} = 1$, we add the inequalities (and corresponding edges)

$$t^s_f \leq t^s_w + \Delta \tag{21}$$
$$t^{s+1}_f \geq t^s_w, \tag{22}$$

if $y^s_{fw} = 1$, we add the inequality

$$t^{s+1}_f \leq t^s_w, \tag{23}$$

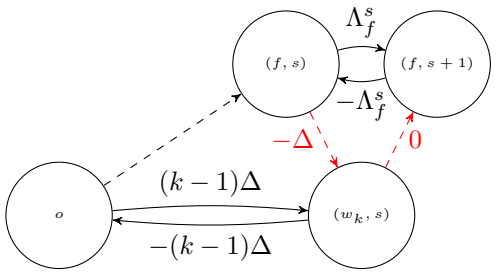

Figure 5. Sub graph of the route node graph. This subgraph encodes the fact that flight f is in sector s in window w_k.

and if $y^s_{wf} = 1$, we add

$$t^s_f \geq t^s_w + \Delta. \tag{24}$$

We are assuming, without loss of generality, a fixed window size Δ. In order to use a variable window size Δ^s_w, simply modify (20) as follows

$$t^s_{w_i} = t_o + \sum_{j=1}^{i-1} \Delta^s_{w_j}, \tag{25}$$

and replace Δ with Δ^s_w in (21) and (24). Note that Δ^s_w 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

$$y^s_{fw} + y^s_{wf} + x^s_{fw} = 1, \qquad f \in F, w \in W, s \in S. \tag{26}$$

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^w_s be the capacity for window w and sector s, then

$$\sum_{f \in F} x^s_{fw} \leq c^w_s, \qquad s \in S, w \in W. \tag{27}$$

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

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

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 colums), 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.

		Type 1A						**Type 1B, 10 seconds time window**						**Type 1B, 1 minute time window**					
$\|F\|$	c_s	**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 contraints, 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.

	Type 1B, 10 minutes time window					**Type 1B, 10 minutes time window**					**Type 1B, 10 minutes time window**				
$\|F\|$	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. Degrmont, "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.