Scheduling Aircraft to Reduce Controller Workload

Joondong Kim[1], Alexander Kröller[2], Joseph S. B. Mitchell[1] and Girishkumar R. Sabhnani[3]

[1] Applied Mathematics and Statistics, Stony Brook University
[2] IBR, Algorithms Group, Braunschweig University of Technology
[3] Computer Science, Stony Brook University

Abstract. We address a problem in air traffic management: scheduling flights in order to minimize the maximum number of aircraft that simultaneously lie within a single air traffic control sector at any time t. Since the problem is a generalization of the NP-hard no-wait job-shop scheduling, we resort to heuristics. We report experimental results for real-world flight data.

Keywords: Air Traffic Management, trajectory scheduling, flight plan scheduling, no-wait job shop.

1 Introduction

In the air traffic control system, the volume of airspace in the altitude range that aircraft utilize is partitioned into a set of *sectors*. We consider the set of all trajectories flown between city pairs. Any one trajectory is modeled as a polygonal path, with each vertex (*way point*) being specified by a point, (x, y, z, t), in space-time. For a given set of sectors and a given set of trajectories, we can compute the *occupancy count*, $n_\sigma(t)$, of a sector σ at any time t. For purposes of air traffic control, it is important that $n_\sigma(t)$ not be "too large"; often the occupancy count is compared with the *Monitor Alert Parameter* (MAP) value of the sector σ, which is related to the "capacity" of the sector. Depending on the timing and routing of the flights, though, the MAP values of certain congested sectors are often predicted to be exceeded (if current flights remain on filed flight plans), resulting in the rerouting of aircraft to avoid those sectors that are anticipated to be at or near full capacity during some period of time.

We consider the following scheduling problem: For a given set of trajectories and a given sectorization of airspace, determine alternate departure times "close" to the originally scheduled times so that the modified trajectories result in minimizing $\max_{\sigma,t} n_\sigma(t)$, the maximum occupancy count of a sector over a time window of interest.

J. Clausen, G. Di Stefano (Eds): ATMOS 2009
9th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2009/2144

2 Problem Statement

Formally, the Min-Max Sector Workload Problem (MMSWP) is defined as follows. We are given a set Σ of *sectors* and a set Θ of periodic *flight plans*. The common period of all plans is T, e.g., $T = 24$ hours. Corresponding to each flight plan θ is a sequence $\Sigma_\theta = (\sigma_{\theta,1}, \sigma_{\theta,2}, \ldots)$ of the sectors it visits, where $\sigma_{\theta,k} \in \Sigma$, $\forall k$. Flight plan θ also has an associated *departure time* $d_\theta \in [0, T)$, and for each sector $\sigma_{\theta,k}$ it has an associated *dwell time*, $t_{\theta,k}$ (length of time in sector).

Assuming a flight θ departs daily with a delay of Δ_θ, it will therefore be in sector $\sigma_{\theta,k}$ during the intervals

$$I_\theta(\sigma_{\theta,k}, \Delta_\theta) := \left[\sum_{\ell<k} t_{\theta,\ell}, \sum_{\ell\le k} t_{\theta,\ell}\right) + d_\theta + \Delta_\theta + T\mathbb{Z}. \tag{1}$$

Therefore, at time $t \in [0, T)$ (and also $t + zT$ for any $z \in \mathbb{Z}$), a total of

$$n_\sigma(t) := |\{\theta \in \Theta : t \in I_\theta(\sigma, \Delta_\theta)\}| \tag{2}$$

flights will be in sector $\sigma \in \Sigma$.

Our goal is to find delays $(\Delta_\theta)_{\theta\in\Theta}$ to minimize the overall maximum occupancy count, $\max_{\sigma,t} n_\sigma(t)$. The delays are constrained to be within the range $[0, D]$ for parameter D. Note that additionally allowing flights to leave early, i.e., $\Delta_\theta < 0$, does not change the problem due to the periodicity of flight plans: A delay range $[-a, b]$ is equivalent to $[0, a + b]$, for $a, b > 0$. Therefore, we just consider the problem where $\Delta_\theta \ge 0$.

3 Job-Shop Scheduling and Related Work

No-wait job-shop scheduling is defined as follows (see [5]): We are given a set of m machines and a set of n jobs that have to be processed on these machines. For each job i, we are given a sequence r_{ik} indicating that job i has to be processed on the kth machine. Additionally, we are given the matrix p_{ij} ($1 \le i \le n, 1 \le j \le m$), stating the processing time of job i on machine j. Furthermore, the following constraints hold:

- *Sequence*: Each job must be processed in order of its operations and no interruption (preemption) of an operation is allowed.
- *Synchronicity*: No job can be processed by two machines at the same time and no machine can process two jobs at the same time.
- *No-wait*: There must be no waiting time between two consecutive operations of the same job.

When there is no constraint on the maximum delay, i.e., $D \ge T$, our problem is equivalent to "no-wait job-shop scheduling". We represent each flight plan as a job and each sector as a machine. We seek to minimize *makespan*, i.e., the smallest time in which all jobs can be processed, where no two jobs can be

on the same machine at the same time. The no-wait constraint ensures that, once started, a job can neither be delayed between machines nor suspended while being processed on one. An optimal solution to the job-shop problem with makespan M can be converted trivially to a flight plan solution with maximum occupancy $\lceil M/T \rceil$. Vice versa, an algorithm for flight plan scheduling also solves job-shop by finding the largest λ for which a flight plan with all processing times scaled by λ can be scheduled with maximum occupancy 1. This can be achieved using binary search.

Lemma 1. *Minimizing makespan in the no-wait job-shop scheduling problem is polynomially equivalent to the Min-Max Sector Workload Problem (MMSWP).*

No-wait job-shop scheduling has been studied in several papers; see, e.g., [8, 10, 11, 9, 7]. Bansal et al. [1] give a $PTAS$ for a special case of the problem and show hardness of approximation for another case. Karger et al. [6] provide a survey of scheduling algorithms, defining the various terms and known results for some of the basic problems. Since the job-shop problem is NP-hard, so is the MMSWP, by Lemma 1.

Ariano et al. [3] formulate train scheduling as a job shop problem with no-store constraints. Bertsimas et. al [2] solve an optimal combination of flow management actions, including ground holding, rerouting, speed control and airborne holding on a flight-by-flight basis.

4 Simplified Cases

In this section, we examine some special cases of the problem. In all the cases here, we consider $D = T$, so that there are no maximum delay constraints.

4.1 One-Sector Problem

In the simplest of cases, there is only sector σ_0 and hence all the flight plans just define the time interval the flight remains in this sector. For all $\theta \in \Theta$, $\sigma_{\theta,1} = \sigma_0$.

If we remove periodicity of flight plans, i.e. put a constraint $d_\theta + \Delta_\theta + t_{\theta,1} \leq T$ hours for each flight θ, the optimal re-scheduling problem of minimizing the *max-workload* exactly maps to the bin-packing problem, which is known to be hard (by a reduction from set partition) and and to have an asymptotic PTAS [4][4].

If we consider periodic flight, then the *one-sector* problem has a trivial solution given by assigning delay to make flights back to back. This gives a max-workload of $\lceil \sum_{\theta \in \Theta} t_{\theta,1}/T \rceil$.

[4] An asymptotic PTAS is an algorithm that, given $\epsilon > 0$, produces a $(1+\epsilon)$- approximate solution provided $OPT > C(\epsilon)$ for some function C, and runs in time polynomial in n for every fixed ϵ.

4.2 Two-Sector Problem

The extension of the problem to two sectors, with a periodic schedule of flights, seems like an interesting special case to understand the complications associated with the *no-wait* constraint and also the periodicity of the schedules. It is much easier to understand the *two-sector* problem by considering its exact equivalent below.

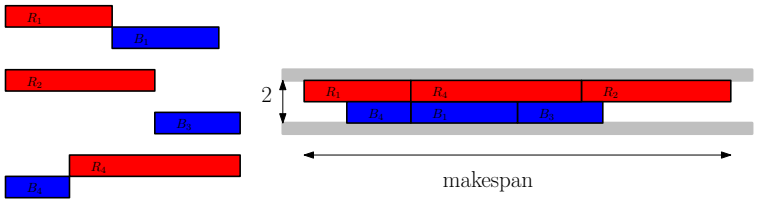

Fig. 1. Left: 4 kinds of blocks. Right: The tight-fitting in the groove of size 2.

Consider Figure 1. Let A, B be the sectors. The red rectangles indicate the time interval of flights in A and the blue rectangles indicate intervals in B. Red to the left of blue indicates that flight starts in A and single red rectangle indicates the flight is only in A. Thus, the MMSWP corresponds to packing these blocks of rectangles as tightly as possible in the groove of width 2, constraining that red rectangles strictly remain in the upper row, blue rectangles strictly remain in the lower row and none of the rectangles overlap.

It turns out that periodicity does not really help for this case, as this version of the problem also turns out to be *NP-complete* by reduction from 3-*PARTITION PROBLEM*.

Theorem 1. *The MMSWP within* 2 *sectors is NP-Complete.*

Proof. $3m$ numbers $a_1, a_2, \ldots, a_{3m}$ are given for a 3-PARTITION PROBLEM instance P. All of these number are between $B/4$ and $B/2$, where mB is the total sum of $a_1, \ldots, a_{3m}$. We show the optimal solution of minimizing workload overall sectors gives us the solution of this problem.

Let's construct the MMSWP problem instance corresponding given input m, B, and a_i's. There are two sectors σ_1 and σ_2. Let time horizon T be $(mB + m)$. For given numbers a_i where $i \in \{1, \ldots, 3m\}$, we generate flights θ_i which visits only σ_1 with staying time a_i, i.e., $\Sigma_{\theta_i} = (\sigma_1)$ and $t_{\theta_i,1} = a_i$ for $i \in \{1, 2, \ldots, 3m\}$. And we prepare additional m flights $\theta_{3m+1}, \ldots, \theta_{3m+m}$ which visit σ_2 for time $(B + 1)$ and then σ_1 for 1. i.e, $\Sigma_{\theta_j} = (\sigma_2, \sigma_1)$ and $t_{\theta_j,1} = (B + 1), t_{\theta_j,2} = 1$ for $j \in \{3m + 1, \ldots, 3m + m\}$.

Then, we claim that if we minimize maximum workload over all sectors for this problem as 1, then we are able to solve given P.

In order to make workload as 1 for σ_2, we have to arrange $\theta_{3m+1}, \ldots, \theta_{3m+3}$ back-to-back like dark-gray blocks in Figure 2. Then there are m intervals with

length B in σ_1. Now finding a placement of $\theta_1, \ldots, \theta_{3m}$ (light gray blocks in Figure 2) to make workload of σ_1 as 1 is finding a partition of $\{a_1, \ldots, a_{3m}\}$ such that each sum is exactly B.

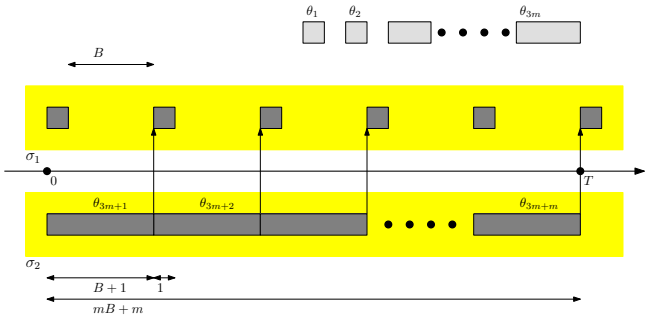

Fig. 2. 2 sectors workload problem construction for given 3-Partition problem instance

5 Algorithms

In this section, we present heuristics to solve the MMSWP.

5.1 Shifting

Starting with the original flight schedule, we pick the sector with worst max-workload (in case of tie check each one of them), and look at the time interval where the max-workload is worse. All the flights present in the sector in that time interval are considered for re-scheduling (shifting) and the one which gives the "best" improvement is selected greedily. The goodness of a shift is judged by its effect on the workload vector which stores the workloads of all sectors in the sorted order. The flight whose re-scheduling gives the best improvement in lexicographic ordering of the workload vector is selected (in case of ties, we pick the flight which has the least difference in the re-schedule time and the original schedule). The process is repeated till all shifts at a given iteration worsen the workload vector. (Note that shifts keep taking place even when the workload vector remains same).
We constrain the greedy shifting to be of the following three kinds:

- Right Shift - The flights are only allowed to be postponed.
- Left Shift - The flight are only allowed to be preponed.
- Short Shift - The decision of postpone/prepone is decided by the amount of shift, and the shorter one is picked.

It is possible to get into loop if we allow shifts in both directions. In our experiments, we only use right shifts to finish algorithm certainly. Since we allow shifts without strict workload vector improvement, all shifts after the last workload vector change are restored when the algorithm is finished.

We also devise an incremental heuristic, in which flights are added one by one (in a random order). With each new flight addition, we run complete experiment of a shift heuristic considering all the flights previously added along with this one.

5.2 Randomized Rounding

The *randomized rounding* algorithm solves a linear problem formulation whose variables describe a probability distribution for each flight plan. Then, a solution is generated by drawing delays from these distributions.

We evenly divide the interval $[0, D]$ into a discrete set of delays $\{0 = d_0, d_1, \ldots, d_m = D\}$. Also we slice the 24h-period T into n pieces $\{0 = t_0, t_1, \ldots, t_n = T\}$.

For each flight θ, the linear formulation has a variable $x_\theta(d_i)$ for each $d_i, 0 \leq i \leq m$. The interpretation (in terms of the finally assigned delay Δ_θ) is

$$x_\theta(d_i) = \Pr[\Delta_\theta \geq d_i] \ .$$

So the $x_\theta(\cdot)$ define a probability function on $[0, D]$ for every flight (the density is constant within each interval $[d_i, d_{i+1})$, that is, the distribution is uniform within each interval). To make sure the $x_\theta(d_i)$ define a proper probability distribution, we use the constraints

$$1 = x_\theta(d_0) \geq x_\theta(d_1) \geq \cdots \geq x_\theta(d_m) = 0.$$

This means the probability that a flight delay is in the range $[d_i, d_j]$ is $x_\theta(d_i) - x_\theta(d_j)$, so the probabilities are nicely encoded in the formulation. Note that

$$\Pr[\text{flight } \theta \text{ is in sector } \sigma \text{ at time } t]$$

is a linear term in the $x_\theta(\cdot)$ variables. To see this, translate t into a range $[\underline{\Delta}_\theta, \overline{\Delta}_\theta]$ of delays where a flight would start to be in σ at t. The probabilities are then:

- Some of the first interval with $d_i \leq \underline{\Delta}_\theta \leq d_{i+1}$, that is,

$$\Pr[\theta \text{ is in } \sigma \text{ at } t \ , \ \Delta_\theta \in [d_i, d_{i+1})] = \frac{d_{i+1} - \underline{\Delta}_\theta}{d_{i+1} - d_i}(x_\theta(d_i) - x_\theta(d_{i+1})) \ .$$

- All of the intervals $\underline{\Delta}_\theta \leq d_i \leq \ldots d_{i+1} \leq \overline{\Delta}_\theta$, in a similar fashion.
- Some interval part around $\overline{\Delta}_\theta$, again analogous to the first case.

By adding the cases, one can see how $\Pr[\theta \text{ is in } \sigma \text{ at } t]$ is a linear term with up to four coefficients. Obviously there are a number of special cases when $[\underline{\Delta}_\theta, \overline{\Delta}_\theta] \not\subseteq [0, D]$; these are easy to resolve and left out in this presentation. So we can now describe the expected load of sector σ at time t by the linear term

$$\mathrm{E}[\text{number of flights in } \sigma \text{ at time } t] = \sum_{\theta \in \Theta} \Pr[\theta \text{ is in } \sigma \text{ at } t].$$

Hence, we solve the following LP:

$$\begin{aligned}
&\min\ C \\
&\text{s.t.}\ \ \text{E[number of flights in } \sigma \text{ at time } t] \le C \quad \forall \sigma \in \Sigma, t \in \{T_o, \ldots, T_n\} \\
&\qquad 1 = x_\theta(d_0) \ge x_\theta(d_1) \ge \cdots \ge x_\theta(d_m) = 0 \quad \forall \theta \in \Theta\ ,
\end{aligned}$$

which gives us a probability distribution for each Δ_θ, so we now generate actual Δ_θ values following these distributions.

An interesting variant arises when we add integrality constraints to the LP, as this forbids smearing flights over many delay intervals. As the resulting IPs are typically impossible to solve within reasonable time, we employ a different strategy: First, the LP-based heuristic is run. We identify the most crowded sectors, and add integrality constraints for tracks passing these sectors. At the same time, we vary n and m for different sectors and tracks, such that the crowded sectors get a more detailed formulation than the others.

6 Lower Bounds

6.1 A Simple Bound

The optimal one sector solution for a sector σ (refer to Section 4.1), for $D = T$, independent of any other sector, is a naive lower bound to its max-workload attained by any scheduling, for any D. Thus, we can optimize each sector individually, and pick the maximum value over all sectors, to obtain a lower bound on the workload attained by an optimal scheduling.

6.2 Linear Programming

The second lower bound algorithm is based on the randomized rounding algorithm. Assume that all the $x_\theta(\cdot)$ are binary, i.e., 0 or 1 (see Section 5.2 for details). If now $x_\theta(d_i) - x_\theta(d_j) = 1$, then flight θ will have a delay $\Delta_\theta \in [d_i, d_j]$.

For a track $\theta \in \Theta$, a sector $\sigma \in \Sigma$ and a time t, we again compute the interval $[\underline{\Delta}_\theta, \overline{\Delta}_\theta]$ of delays for θ under which θ will be in σ at t. Then we determine the smallest $d_i \ge \underline{\Delta}_\theta$ and the largest $d_j \le \overline{\Delta}_\theta$. Then, when $x_\theta(d_i) - x_\theta(d_j) = 1$, the flight will be in σ at t. So define $g_\theta(\sigma, t) := x_\theta(d_i) - x_\theta(d_j)$.

The following IP charges 1 towards the maximum capacity C when a track is guaranteed to be in σ at t:

$$\begin{aligned}
&\min\ C \\
&\text{s.t.}\ \ \sum_{\theta \in \Theta} g_\theta(\sigma, t) \le C && \forall \sigma \in \Sigma, t \in \{T_o, \ldots, T_n\} \\
&\qquad 1 = x_\theta(d_0) \ge x_\theta(d_1) \ge \cdots \ge x_\theta(d_m) = 0 && \forall \theta \in \Theta \\
&\qquad x_\theta(d_i) \in \{0, 1\} && \forall \theta \in \Theta, i = 0, \ldots, m
\end{aligned}$$

The optimal solution to this IP is a lower bound to the original problem. For efficiency reasons, we do not solve this IP directly, but rather its LP relaxation, which is obtained by dropping the integrality constraint.

7 Results

We use real-world flight track data and sector data from the National Airspace System (NAS). The data, as shown in Table 1, is divided into 5 sets depending on the number of sectors. The *alt-range* defines the range of altitude for the air-traffic in the sectors. The high-altitude sectors typically have *alt-range* $24,000$ feet and above. **Set1**, **Set2** and **Set3** consider flight tracks for the entire 24 hour time period while **Set4** considers only the flights that overlap a 4 hour time window. Note that the flight times may start or end outside the 4 hour time window. Also, **Set4** includes all the sectors spanned by these flights, thus having high-altitude sectors, low-altitude sectors and some sectors from Canada as well.

	No. of Sectors	Alt-Range	Flights	Time Window
Set1	5	$\geq$ 24k feet	1904	$0-24$ hrs
Set2	18	$\geq$ 24k feet	3063	$0-24$ hrs
Set3	57	≥ 0 feet	12123	$0-24$ hrs
Set4	1281	Different	11986	$14-18$ hrs
Set5	16	$\geq$ 24k feet	4994	$0-24$ hrs

Table 1. Summary of data sets used for experimentation.

Set5 (random data) consists of a 300×300 nautical miles square region divided into 16 sectors in the form of a square grid. Then, 64 (uniform) random cities were generated such that 10% of cities had weight 10, 15% had weight 5, and the remaining had weight 1. In total, 4994 random flights were generated between (weighted uniform) randomly chosen city pairs, with each city having probability of selection proportional to its weight. The departure-time of a flight was (uniform) randomly generated between $0-24$ hours. The (constant) speed of an aircraft was modeled as a (uniform) random variable between 200 and 600 nautical miles per hour. The arrival-time of a flight was calculated from the departure time, the speed of the aircraft, and the distance between the cities in the pair. An additional constraint was added that no two aircraft depart from (or arrive) at a city within 1 minute of each other. A visualization of data sets **Set1**, **Set2** and **Set5** can be seen in Figure 3.

We implemented our algorithms and ran them on the five data sets. For the LP-based algorithms, we used CPLEX 10.0 on a 3.0 GHz Linux machine. We solved each instance using a few parameter sets, varying the number of discretizations in delay (i.e., m) and daytime slices (i.e., n). The most often used values of $m = 30$ and $n = 720$ correspond to having one variable per two minutes of delay and one constraint for every other minute of the day. We imposed a runtime limit of 60 minutes on the algorithm. Table 2 describes these runs and lists the according algorithm runtimes. Runtimes for the other heuristics are not listed, as they always finish within a few seconds.

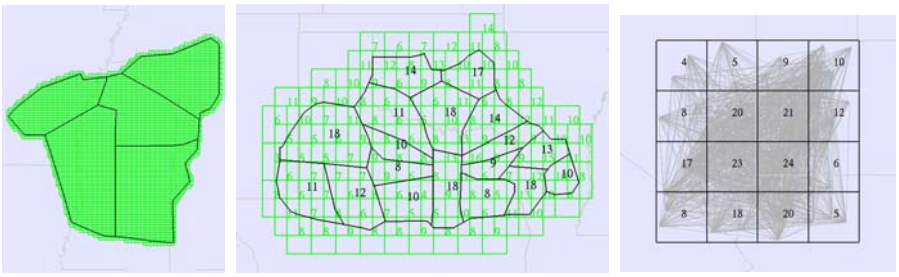

Fig. 3. Left: Set1 sectors and the underlying square grid (and shifted square grid) cover (grid resolution: 0.1x0.1); Center: Set2 sectors and grid cover (1x1). Right: Set5 (randomly generated) flight tracks with the underlying sectors. The numbers in the sectors indicate the max-workload counts for the used flight schedules.

	Set1			Set2			Set3			Set4			Set5		
	m	n	Time	m	n	Time	m	n	Time	m	n	Time	m	n	Time
LP Lower	30	720	1:20	30	720	1:50	30	720	9:10	60	1440	17:19	30	720	10:26
MIP Lower	30	720	3:04	–	–	–	12	288	10:18	12	288	14:44	–	–	–
Rand. Rounding	30	720	22:24	12	288	1:05	12	288	30:07	30	720	57:11	12	288	10:18
MIP Rounding	12	288	0:28	12	288	0:33	12	288	56:17	12	288	17:30	12	288	5:13

Table 2. Details for LP-based heuristics, showing the discretization granularity and total algorithm runtimes in minutes.

	Set1			Set2			Set3			Set4			Set5		
	Max	Mean	Var	Max	Mean	Var	Max	Mean	Var	Max	Mean	Var	Max	Mean	Var
Original plan	22	18.00	6.80	18	12.83	12.25	38	21.56	36.70	58	7.67	37.88	24	13.00	46.13
Right Shift	18	16.40	1.04	14	11.11	3.99	31	20.77	26.27	47	7.61	36.35	19	11.75	29.01
Incr. Right Shift	15	13.80	0.96	**12**	10.17	2.25	**26**	18.75	16.40	**39**	7.51	34.50	**17**	10.81	20.66
Rand. Rounding	**14**	13.40	0.24	14	11.67	4.00	28	22.94	19.50	42	8.04	40.50	19	12.50	25.00
MIP	15	14.40	0.24	14	11.22	4.73	28	23.47	16.18	43	8.22	44.90	19	12.50	30.13
	Naive	LP	IP	Naive	LP	IP	Naive	LP	IP	Naive	LP	IP	Naive	LP	IP
Lower Bound	6	**9**	**9**	5	**8**	–	16	**20**	14	12	**31**	22	**13**	11	–

Table 3. Workload statistics of algorithms. Max: Maximum Workload, Mean: Mean of workload, Var: Variance of workload

	Set1 (1904 flt)			Set2 (3063 flt)			Set3 (12123 flt)			Set4 (11986 flt)			Set5 (4994 flt)		
	Max	Total	Avg	Max	Total	Avg	Max	Total	Avg	Max	Total	Avg	Max	Total	Avg
Right Shift	6	46	1	9	5:25	1	17	5:18	1	53	12:53	4	7	3:8	1
Incr. Right Shift	49	2:00:46	4	52	3:16:21	6	60	18:21:7	6	60	14:22:54	17	54	4:18:5	4
Rand. Rounding	60	13:22:24	10	60	13:06:48	6	60	35:18:15	4	58	50:10:59	6	55	59:16:33	17
MIP	60	14:21:48	12	60	15:21:42	7	60	37:10:59	4	55	90:00:38	11	55	60:05:50	17

Table 4. Time shift statistics of various methods. Max: Max shift, Total: Sum of absolute value of shift, Avg: Average of absolute value of non-zero shifts. (format 14:21:48 means 14 days 21 hours 48 minutes)

Table 3 shows the comparison of max-workload statistics of the given flight plans, the heuristic solutions and the LP based methods. The maximum allowable shift to any flight schedule was constrained to be 1 hour in all methods. The discretization of time for LP/IP methods is 1 minute. The results show a considerable improvement over the workloads of each sector arising due to the original flight schedules. Even the variance values have gone down significantly, indicating more balance of workload across sectors. In particular, the incremental shift heuristic seems to out-perform all the other methods. Note that the shifting heuristics do not discretize the time like LP/MIP methods. The '-' values in Table 3 refer to experiments for which no solution was found during more than a week of running time.

Table 3 also shows the lower bound calculations for the 5 sets. The best solutions are still not close to the computed lower bounds, but we believe they are very close to optimal solutions. Future work will specifically aim to improve the lower bounds.

Table 4 shows the statistics of the amount of time shifts from the original schedule. *Max* indicates the maximum shift in any flight schedule, *Total* indicates the sum of absolute values of shifts, and the *Avg* gives the average time shift of all flights with non-zero shifts. The value of *Total* in the case of the right shift heuristic is noticeably small compared to other methods, possibly because of early termination due to reaching a local minimum. Also, the average time shift is seen to be low for all the methods, suggesting that we can get considerable improvements in workloads with reasonable modification to the schedules.

8 Other Workload Considerations

Apart from the *max-workload* of a sector, there are other workload issues which are significant from the controller perspective. One of them, usually referred to as *coordination* workload, deals with the hand-offs between controllers when an aircraft moves from one sector to the other. Another critical issue is the *conflict resolution* workload, which is related to monitoring the aircraft when they are expected to be simultaneously present at (or near) the same geographic point (a "conflict point"). Note that even if two aircraft are flying at different altitudes, at the conflict point, they demand special attention of the controller.

While re-scheduling flights has no effect on the *coordination* workload, it can favorably affect the *conflict resolution* workload, by reducing the number of conflict points. It is easy to incorporate conflict resolution workload in the model, as we now discuss.

We sub-divide the region (spanned by the sectors) into (reasonably) small size cells and compute the max-workload in each cell separately. If the size of the cell is small, a high max-workload cell corresponds to a conflict point, where multiple aircraft are in close proximity simultaneously. We add these cells as new (artificial) sectors to the data set and try to minimize their workload vector separately, thereby (possibly) decreasing the number of conflict points.

The shifting heuristic is now modified to be a two-step procedure. The first step considers the overall maximum value of the max-workload across all cells to be a constraint: The aircraft are re-scheduled to improve the workload vector of the sectors, as before, while keeping the workloads in all cells below a specified W_c. In the second step, the roles of sectors and cells are reversed: The optimized maximum value of the workload of the sectors is treated as a constraint, and the aircraft are re-scheduled with the objective of improving the workload vector of the cells.

For experimentation, these cells come from a uniform (square) grid and a shifted uniform grid as shown in Figure 3 covering the region spanned by the sectors. Two different side lengths of square grid cells are used, 0.1×0.1 and 0.2×0.2 (unit latitude/longitude degrees). In Set1, Set2 and Set5, 1 degree corresponds to somewhere in the range of $35-60$ nautical miles. Table 5 shows the results of the workload improvements with the cell constraints. We observe that the max-workloads of the sectors still improve, compared with the original (18 v/s 22 for Set1), while the number of conflict points are considerably decreased (see Figure 4). For Set1, after scheduling there are no grid cells with workload 4, while the number of cells with workload 3 has also decreased by more than 90%.

Grid Size	Set1 (Given SMax: 22)					Set2 (Given SMax: 18)					Set5 (Given SMax: 24)				
	Given		Shifted			Given		Shifted			Given		Shifted		
	GMax	GMean	SMax	GMax	GMean	GMax	GMean	SMax	GMax	GMean	GMax	GMean	SMax	GMax	GMean
0.1×0.1	4	1.670	18	3	1.604	4	1.467	14	4	1.478	11	1.609	19	8	1.598
0.2×0.2	5	2.446	18	4	2.356	5	2.105	14	4	2.083	14	2.271	19	10	2.243

Table 5. Results of Right-Shift heuristic with additional grid constraints. SMax: Sector Max, SMean: Sector Mean, GMax: Grid Max, GMean: Grid Mean.

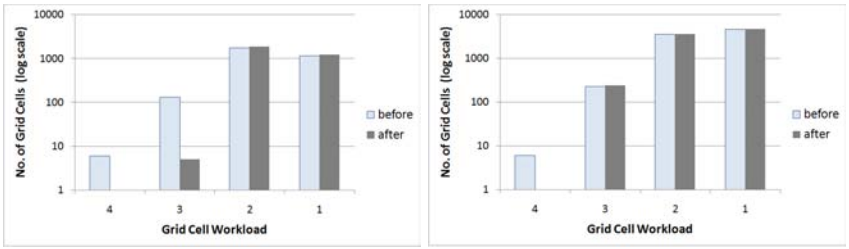

Fig. 4. Left: Set1 grid cell max-workloads; Right: Set2 grid cell max-workloads (before and after scheduling, for grid size 0.1×0.1)

9 Conclusion

We presented a periodic flight plan scheduling problem, proved it to be NP-hard, and proposed heuristics for which we reported experimental results on real-world data. The results show a considerable workload improvement over the originally scheduled flight times and come at low computational cost. The reduction in the number of conflict points was also impressive. Future work will specifically aim to improve the lower bound, as we believe that the heuristically produced solutions are already almost optimal. Also, we are interested in combining re-routing with re-scheduling to improve further the workloads.

Acknowledgements. The data used for the experiments was provided by Metron Aviation. We thank Michael Bender and Bob Hoffman for helpful discussions. This work was partially supported by NSF (CCF-0528209, CCF-0729019), NASA Ames, and Metron Aviation.

References

1. N. Bansal, M. Mahdian, and M. Sviridenko. Minimizing makespan in no-wait job shops. *Math. Oper. Res.*, 30(4):817–831, 2005.
2. D. Bertsimas, G. Lulli, and A. Odoni. The air traffic flow management problem: An integer optimization approach. In *13th International Conference on Integer Programming and Combinatorial Optimization, IPCO 2008 Bertinoro*, volume 5035, pages 34–46, May 2008.
3. A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, December 2007.
4. W. F. de la Vega and G. Lueker. Bin packing can be solved within $1 + \epsilon$ in linear time. *Combinatorica*, 1(4):349–355, 1981.
5. J. M. Framinan and C. Schuster. An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach. *Comput. Oper. Res.*, 33(5):1200–1213, 2006.
6. D. Karger, C. Stein, and J. Wein. Scheduling algorithms. *CRC Handbook of Computer Science*, 1997.
7. P. M. Lennartz. *No-Wait Job Shop Scheduling, a Constraint Propagation Approach.* PhD thesis, UU Universiteit Utrecht, Netherlands, 2006.
8. A. Mascis and D. Pacciarelli. Job shop scheduling with blocking and no-wait constraints. *Eur J. Oper. Res.*, 142:498–517, 2002.
9. C. J. Schuster. No-wait job shop scheduling: Tabu search and complexity of subproblems. *Mathematical Methods of Operations Research*, 63(3):473–491, July 2006.
10. C. J. Schuster and J. Framinan. Approximative procedures for no-wait job shop scheduling. *Oper Res Lett*, 31:308–318, 2003.
11. G. J. Woeginger. Inapproximability results for no-wait job shop scheduling. *Oper. Res. Lett.*, 32:320–325, 2004.