JOURNAL OF AIRCRAFT
Vol. 49, No. 1, January–February 2012

Sequential Dynamic Strategies for Real-Time Scheduling of Terminal Traffic

Heming Chen* and Yiyuan J. Zhao†
University of Minnesota, Minneapolis, Minnesota 55455

DOI: 10.2514/1.C031503

This paper presents dynamic strategies for the integrated scheduling and runway assignment of both arrival and departure traffic over an airport. While a static scheduling scheme handles traffic over a specified planning horizon simultaneously, sequential dynamic schedulers divide the planning horizon into a series of smaller scheduling windows and apply a static scheduling scheme sequentially over each window. Dynamic scheduling strategies are desirable for obtaining real-time solutions of continual traffic streams and for taking advantage of updated traffic information. In this paper, a multiple-point scheduling framework is used in which scheduling locations include runway thresholds as well as fixes over the terminal airspace and gates on the airport surface. Integrated static scheduling of both arrival and departure traffic is formulated as mixed-integer linear programming. Solution variables include scheduled times of arrival at the multiple scheduling locations and aircraft sequences at merge points. Aircraft route assignments for both ground and airborne traffic are also included as discrete solution variables, from which optimal runway assignments can be determined. Then, different dynamic strategies with either overlapping or nonoverlapping scheduling windows are developed and compared. Induced constraints for ensuring sufficient separations among traffic in neighboring windows are discussed. The John F. Kennedy International Airport is used as the example in extensive numerical solutions to evaluate the computational speeds and scheduling performances of different dynamic strategies.

Nomenclature

C	=	cost of delay
i, i_1, i_2	=	arrival indices, where $1 \le i, i_1, i_2 \le N_{\text{arr}}$
j, j_1, j_2	=	departure indices, where $1 \le j, j_1, j_2 \le N_{\text{dep}}$
K	=	total number of common sections between two routes
k	=	common section index, where $1 \le k \le K$
$\mathcal{M}$	=	large positive real number
N_{arr}	=	total number of arrivals
N_{dep}	=	total number of departures
P	=	total number of arrival scheduling points on an arrival route
p, p_1, p_2	=	scheduling point indices on arrival routes, where $1 \le p, p_1, p_2 \le P$
Q	=	total number of departure scheduling points on a departure route
q, q_1, q_2	=	scheduling point indices on departure routes, where $1 \le q, q_1, q_2 \le Q$
R	=	total number of arrival routes
r, r_1, r_2	=	arrival route indices, where $1 \le r, r_1, r_2 \le R$
S	=	total number of departure routes
s, s_1, s_2	=	departure route indices, where $1 \le s, s_1, s_2 \le S$
t	=	scheduled time of arrival
T_{adv}	=	advance time
T_C	=	computational time for one scheduling window
T_L	=	look-ahead time
T_W	=	length of scheduling windows
T_{wait}	=	Waiting time
$\hat{t}$	=	estimated time of arrival

Received 13 May 2011; revision received 11 September 2011; accepted for publication 12 September 2011.

*Graduate Student, Aerospace Engineering and Mechanics; hmchen@aem.umn.edu. Student Member AIAA.

†Professor, Aerospace Engineering and Mechanics; gyyz@aem.umn.edu. Associate Fellow AIAA.

I. Introduction

TO IMPROVE both airport efficiency and safety, it is highly desirable to develop computer systems that can assist air traffic controllers in runway scheduling tasks, such as selecting optimal schedules, sequences, and runway assignments. This is consistent with the goal of achieving safe, efficient, and environmentally friendly airport operations in the Next Generation Air Transportation System (NextGen) [1].

In general, optimal airport operations seek to maximize the capacity, throughput, or aircraft efficiencies, and to minimize environmental impacts through the most efficient use of all relevant airport resources. These resources include gates, taxiways, runways, and the terminal airspace. In addition, both arrival and departure traffic should be scheduled together since they share the use of airport resources. As a result, an integrated runway scheduling solution should produce schedules of runway assignments as well as arrival times at these airport resources for all traffic. This requires combined routing and scheduling of traffic, both in the air and on the ground. Furthermore, meaningful scheduling solutions must satisfy aircraft performance limitations over the airport and terminal airspace, such as transit time limits between two locations. They must also respect practical separation requirements that depend on aircraft weight classes as well as their flight phases, such as descent, climb, and taxi of aircraft pairs.

This above problem or, often, its simplified versions have been traditionally solved with static scheduling methods that handle the relevant traffic in a batch. Besides, single-point scheduling has been extensively studied historically using optimization methods [2,3]. These studies typically assume the runway threshold as the scheduling point. The majority of these works considers arrival traffic only [4–13], whereas a few studies examine both arrival and departure traffic [14–16]. Investigations have also been made on optimal planning and scheduling of aircraft motions on airport surfaces using mixed-integer linear programming (MILP) [17–20]. Keith et al. [21] extend the above works by solving a combined routing and scheduling problem. Furthermore, efforts have also been made to study strategic level aircraft scheduling. In particular, Cheng [22], Bolat [23], and Kim et al. [24] study arrival traffic scheduling and traffic congestion relief using a variety of approaches, such as gate assignment. In solving metroplex scheduling, Capozzi et al. [25]

use a combined optimal routing and scheduling problem for airborne traffic.

In practice, air traffic continually arrive into or leave airports. On the other hand, static algorithms only schedule traffic for which the estimated times of arrival (ETAs) fall within a certain time interval of finite length, which is typically shorter than the length of traffic rush hours at busy airports. Even if the static scheduling interval can be increased with the ever-increasing computing powers, static algorithms are structurally insufficient to accommodate ETA uncertainties that usually grow as the look-ahead time increases in aircraft trajectory predictions. Therefore, for effective and efficient practical operations, dynamic strategies are needed that can schedule continually arriving and varying traffic in real time. In this regard, practical real-time scheduling strategies that incorporate controller preferences have been developed and tested. These strategies often use a combination of optimization techniques, intelligence methods, and operation rules [26–32].[‡]

This paper presents a dynamic multiple-point scheduling framework for the integrated scheduling and runway assignment of both arrival and departure traffic over an airport. This framework periodically calls a static scheduling function as its core. Different divisions of the scheduling windows and inclusions of constraints over neighboring windows constitute different dynamic strategies. Furthermore, a bound on the maximum computational time is explicitly imposed to ensure the timely completion of the solution process. This may result in suboptimal solutions in heavy traffic, but it ensures the availability of feasible solutions for real-time applications. This bound can be adjusted in different applications to obtain the best compromise between computational speeds and scheduling performances.

In the remainder of the paper, a multiple-point static scheduler developed in our previous research [33] is reviewed in Sec. II. The need for dynamic scheduling is then discussed in Sec. III. The structures and different designs of dynamic strategies are also presented. The numerical simulation environment and criteria for evaluating the performances of dynamic strategies are discussed next in Sec. IV, followed by presentations of results in Sec. V and Sec. VI. Performances of using overlapping versus nonoverlapping scheduling windows are compared, and benefits of including runway assignments in the scheduling process are examined. Conclusions are presented at the end.

II. Integrated Static Scheduling

A multiple-point static scheme has been developed for the integrated routing and scheduling of both arrival and departure traffic over the terminal area [33] that constitutes the foundation of the proposed dynamic strategies. Specifically, terminal traffic for which the ETAs are between a certain look-ahead time from now and a specified future time are scheduled simultaneously. The multiple scheduling points include not only the runway thresholds as conventionally done but also fixes and gates for both arrival and departure traffic. Solution variables include discrete route selections, discrete sequences at merge points, and continuous scheduled times of arrival (STAs) at the different scheduling locations. Then, MILP is used to obtain numerical solutions. Runway assignments are automatically determined once routes are selected, because a complete route goes through a specific runway. Bounds are imposed on aircraft transit times both in the air and on the ground between neighboring scheduling points to ensure physically meaningful solutions. Appropriate aircraft separations are maintained that are dependent on not only aircraft pairs but their flight phases. For the convenience of discussions in this paper, key dimensions and variables of the static scheduling problem are summarized in the nomenclatures.

Specifically, the three types of decision variables in the static MILP formulation determine flight arrival times at the multiple scheduling locations, route assignments, and flight sequences:

$$t_{ip} = \text{STA of flight } i \text{ at its } p\text{th scheduling point}$$

$$x_i^r = \begin{cases} 1 & \text{if route } r \text{ is assigned to flight } i, \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ij}^k = \begin{cases} 1 & \text{if flight } i \text{ is ahead of flight } j, \text{ on their } k\text{th common section} \\ 0 & \text{otherwise} \end{cases}$$

In general, different optimization criteria may be selected for the integrated runway routing and scheduling, such as airport capacity, airport throughput, aircraft efficiencies, and environmental impact. Besides, these criteria may be represented by their total values, average values, peak values, or weighted sums. The resulting solution processes are similar for problems with different optimization criteria. For the discussions of dynamic strategies in this paper, a weighted overall flight delay is minimized:

$$\min_{T,X,Z} I = \sum_{i=1}^{N_{\text{arr}}} C_i (t_i^{\text{gate}} - \hat{t}_i^{\text{gate}}) + \sum_{j=1}^{N_{\text{dep}}} C_j (t_j^{\text{fix}} - \hat{t}_j^{\text{fix}}) \tag{1}$$

where T, X, and Z represent the collections of continuous decision variables, discrete route selection variables, and discrete sequence variables, respectively. In this expression, t_i^{gate} and t_j^{fix} are the gate STA and fix STA for arrival i and departure j, respectively, C_i and C_j are the weighting factors for the delays, and $\hat{t}$ represents an ETA. It is assumed that no time advance is used. Therefore, the difference between a STA and its corresponding ETA is defined as the delay.

Proper constraints are essential to ensure the physical feasibility of the solutions. Constraints in the above MILP formulation come from different sources, including aircraft performance limits, runway configurations, ambient conditions, and operational procedures. For example, an aircraft is not able to change its speed or altitude instantaneously, and its speed must stay within a meaningful range. Besides, different aircraft must pass through the same scheduling point at different times in order to maintain sufficient separations. Constraints used by Chen et al. [33] are summarized in Table 1.

In the above scheme, each flight is assumed to travel on a route connecting its arrival fix to a terminal gate or vice versa. Two routes are considered different as long as they have at least one different segment. The complete set of routes constitutes a route network representation of an airport. The concept of the route network is an extension of the model used by Keith et al. [21], who discuss optimal surface aircraft routing and scheduling. Figures 1 and 2 illustrate the route network used in this study. In particular, Chen et al. [33] compare two schemes of route selections for the integrated scheduling solutions. The fix-route algorithm computes the optimal flight sequences and STAs for all flights along their default routes. By contrast, the open-route algorithm allows route changes and produces optimal route assignments in addition to optimal flight sequences and STAs.

While the above MILP formulation represents a fairly comprehensive runway routing and scheduling problem, the static nature limits its use in practical real-time applications. Specifically, the following:

1) In practical operations, especially during rush hours, arriving and departing aircraft over a terminal airspace form continual streams of traffic. The static algorithms, designed to schedule an isolated batch of traffic within a certain time interval, cannot easily handle continual traffic streams. In fact, the choice of the scheduling window size presents a challenge. Too small a scheduling window cannot accommodate the amount of traffic that is closely spaced, whereas too large a scheduling window can make the solution process excessively long.

2) Even if a large scheduling window can be used for a specific application, there is no guarantee that necessary computations can be

[‡]Data available at http://www.aviationsystemsdivision.arc.nasa.gov/research/foundations/index.shtml [retrieved 2011].

Table 1 Constraints in integrated scheduling

Number	Type	Formula
1	Single route	$\sum_{r=1}^{R} x_i^r = 1$
2	Sequencing	$z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq 0$ $z_{i_1 i_2}^k - \frac{1}{\mathcal{M}}(t_{i_2 p_2} - t_{i_1 p_1}) - \mathcal{M}(2 - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \leq 1$
3	Separation	$(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(3 - z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2)$ $-(t_{i_2 p_2} - t_{i_1 p_1}) + \mathcal{M}(2 + z_{i_1 i_2}^k - x_{i_1}^{r_1} - x_{i_2}^{r_2}) \geq \delta(z_{i_1 i_2}^k, w_{i_1}, w_{i_2}, p_1, p_2)$
4	Aircraft performance	$t_{i(p+1)} - t_{ip} + \mathcal{M}(1 - x_i^r) \geq \tau_{\min}(c_i, p)$ $t_{i(p+1)} - t_{ip} - \mathcal{M}(1 - x_i^r) \leq \tau_{\max}(c_i, p)$
5	Additional	$t_{i_1}^{\text{gate}} = t_{i_1 P}^r$ $t_{i_2}^{\text{fix}} = t_{i_2 P}^s$

completed within a specified time frame, especially in heavy traffic. During rush hours at busy airports, the traffic volume can increase drastically. The computational times of a static scheduling algorithm can increase significantly as the volume of traffic increases.

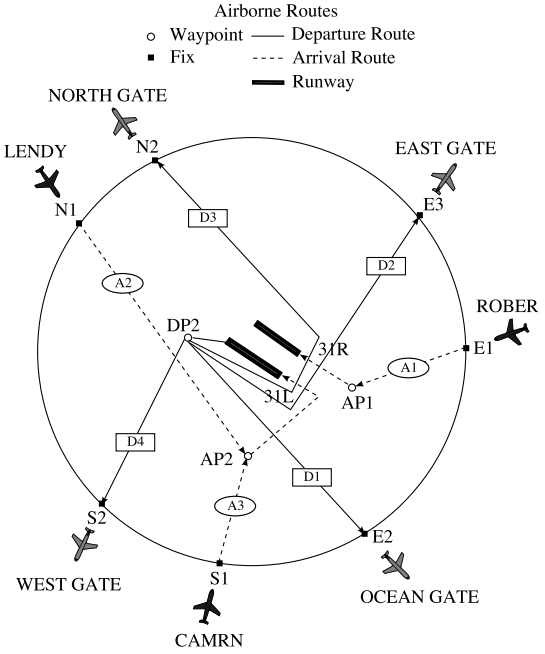

Fig. 1 Airborne routes in simulations (not to scale).

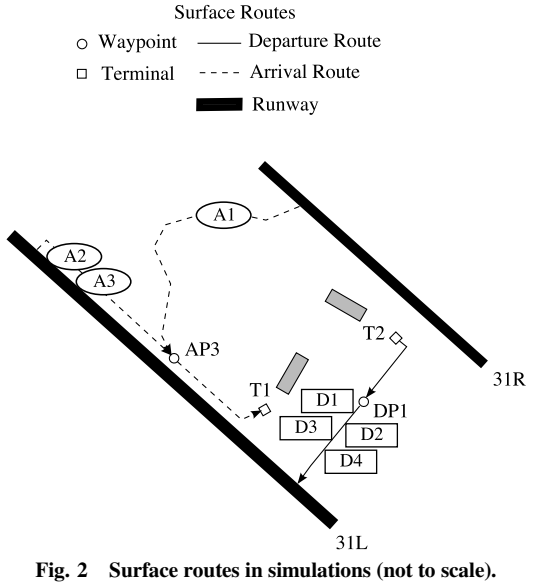

Fig. 2 Surface routes in simulations (not to scale).

3) Finally, optimal schedules are determined based on ETAs that are typically obtained from aircraft trajectory predictions. As the prediction look-ahead time increases, uncertainties in predicted trajectories increase and can limit the length of meaningful planning horizons using static scheduling algorithms. As a result, statically optimal schedules obtained over an overly large planning horizon are no longer reliable. Desirably, the optimal scheduling process should seek to take advantage of the newest traffic information available.

III. Structures of Sequential Dynamic Scheduling

A dynamic scheduling strategy divides the planning horizon that begins at a certain look-ahead time from now and extends indefinitely into the future into a series of smaller time intervals called scheduling windows. It then applies an appropriate static scheduling algorithm sequentially to each window. Sizes of the scheduling windows are selected to make sure that the static algorithms remain computationally efficient. A bound is imposed on the maximum computational time over each scheduling window so that feasible solutions can be obtained in real time, although some solutions may be suboptimal. In addition to the constraints used in the static scheduling algorithms, additional constraints are needed, called induced constraints, to ensure sufficient interaircraft separations among traffic in neighboring windows. As a result, different dynamic strategies amount to different ways of dividing the planning horizon into smaller scheduling windows and establishing proper induced constraints, as well as a proper selection of the bound on the maximum computational time.

There is another challenge in the dynamic strategy design. Because of the multiple-point scheduling scheme, ETAs of a given aircraft at different scheduling points may fall into different scheduling windows. As a result, STAs of the same aircraft at some scheduling points may have been determined over a particular scheduling window, whereas its STAs at other scheduling points may still be open because their corresponding ETAs fall into future scheduling windows. Furthermore, STAs determined in previous scheduling windows may be revised in the current window. Therefore, the design of a dynamic strategy also includes the specification of open scheduling points for a given aircraft over a certain window.

In general, different design choices of a dynamic strategy can be made to improve the scheduling optimality and computational speed,

Table 2 Dynamic strategy parameters

Parameters	Meaning	Options in tests
T_W	Scheduling window size	5–15 min.
T_{adv}	Advance time	100, 75, and 25% of T_w Correspond to 0, 25, and 75% overlaps
T_C	Computational time for each window	Maximum of 90 s
Static algorithm	Computation algorithm to produce optimal schedules	Fix-route algorithm Open-route algorithm

under specific traffic conditions and operational requirements. Table 2 summarizes key components of dynamic scheduling.

For a given dynamic strategy, different solution methods can be used for subinterval scheduling solutions. In this paper, a MILP formulation is used. However, MILP models generally lack theoretical proofs of bounds on computational complexity. Alternative approaches are available in the literature to solve the MILP formulation that have proofs of computational complexity [34,35].

A. Time Relations in Dynamic Scheduling

There are two time concepts in dynamic scheduling: actual time and planning time. Actual time is the current clock time in the real world, whereas a planning time is an instant on the planning horizon that is later than the actual time. These time concepts are illustrated in Fig. 3, in which the vertical axis and horizontal axis represent the actual and the planning time, respectively.

At an actual time t_1, a dynamic strategy begins planning for future traffic that shall arrive at $t_1 + T_L$, where T_L is called the look-ahead time. The dynamic strategy first considers traffic over the scheduling window of $[t_1 + T_L,\ t_1 + T_L + T_W]$, where T_W is the size of the scheduling window. T_C represents the computational time in obtaining solutions over this window. After scheduling solutions for this window are obtained, the dynamic algorithm would wait until actual time t_2. It will then begin scheduling traffic that plans to use the airport over the window of $[t_2 + T_L, t_2 + T_L + T_W]$. This process is then repeated.

B. Choices of Scheduling Window Divisions

There are two general approaches to dividing the planning horizon into smaller scheduling windows: nonoverlapping windows and overlapping windows. Figure 3 illustrates the case when scheduling windows overlap, whereas Fig. 4 illustrates the nonoverlapping scheduling windows. In either case, the advance time between the computations of neighboring scheduling windows is defined as

$$T_{\text{adv}} = t_{n+1} - t_n \tag{2}$$

where t_n is the beginning time of the nth scheduling window and $n = 1, 2, \cdots$. If the scheduling windows do not overlap, we have

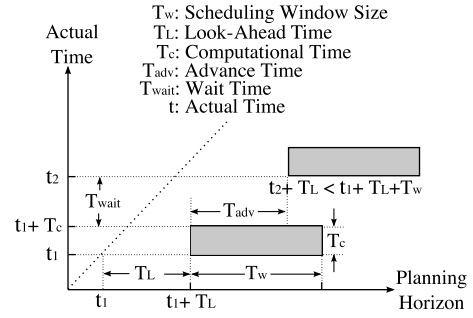

Fig. 3 Timeline with overlapping scheduling windows.

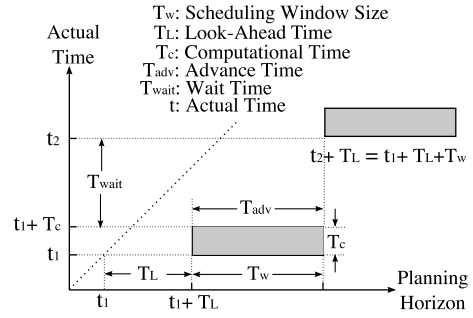

Fig. 4 Timeline with nonoverlapping scheduling windows.

$$T_{\text{adv}} = T_W \tag{3}$$

Otherwise,

$$T_{\text{adv}} < T_W \tag{4}$$

When computations for the scheduling window beginning at t_n are completed, computations for the subsequent scheduling window will not start until t_{n+1}. Assuming that the total computational time for a window is T_C, the waiting time is given by

$$T_{\text{wait}} = T_{\text{adv}} - T_C \tag{5}$$

During the waiting time, the most recent traffic data may be acquired for subsequent computations. T_{wait} also serves as a safety buffer between two computations in case the preceding computation exceeds T_C. In any circumstance, we have

$$T_{\text{adv}} \geq T_C \quad \text{or} \quad T_{\text{wait}} \geq 0 \tag{6}$$

In the absence of traffic data uncertainties, the scheduling window size needs to be selected to strike a balance between scheduling optimality and computational speed. Intuitively, optimization over a larger scheduling window can produce better results than piecewise optimizations over a series of smaller windows. On the other hand, an overly large scheduling window may contain an excessive number of flights, and thus requires a considerable computational time, which can prevent static scheduling algorithms from generating timely solutions. The presence of data uncertainties also practically limits the choices of scheduling window sizes.

Furthermore, proper choices of overlaps between neighboring windows can potentially improve the computational speed, especially when the scheduling window sizes are large. When two adjacent scheduling windows overlap, traffic in the overlapped segment can be optimized in both windows. In most cases, scheduling optimization over the second window only needs to apply small revisions to solutions obtained in the previous window. This often requires less computation efforts. By contrast, two adjacent nonoverlapping windows contain distinct flights. Each of them has to process a large number of new flights, which takes the static algorithm more time to obtain solutions.

In this paper, effects of scheduling window sizes on scheduling performance and computational speed are studied. The differences between nonoverlapping and overlapping windows are also examined.

C. Effect of Multiple Scheduling Points

In a multiple-point scheduling scheme, individual flights are scheduled at several locations over the terminal airspace that include fixes, runway thresholds, and gates. Because of the limited size of a scheduling window, a flight over the terminal air space may span several scheduling windows. As a result, it is possible that ETAs of a given flight at different locations fall into different scheduling windows. This creates a new problem that is absent in the static multiple-point scheduling scheme: Which of the arrival times should be optimized in a given scheduling window?

Figure 5 presents an example of how a dynamic strategy schedules a particular flight, Flight 01, over two scheduling windows. When window 1 is active, all five ETAs of Flight 01 are either in or after window 1. Therefore, we allow the dynamic strategy to adjust all ETAs for the best performance. Correspondingly, five STAs are computed, which are then treated as the new ETAs of Flight 01 in subsequent scheduling calculations. When window 2 becomes the current scheduling window, ETA1 and ETA2 (STA1 and STA2 in window 1, respectively) are located in window 1 and are therefore locked by the dynamic strategy. On the other hand, ETA3 (STA3 in window 1) is now in window 2 and can still be adjusted by the dynamic strategy. Similarly, ETA4 and ETA5 can also be adjusted for the best performance in window 2. In this paper, the dynamic strategy

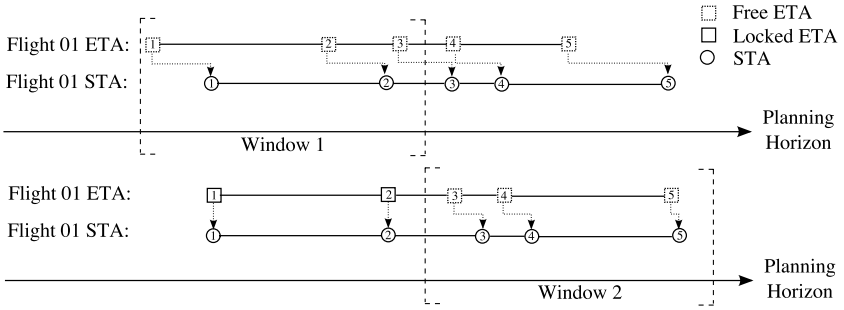

Fig. 5 Multiple-point scheduling over multiple scheduling windows.

locks ETAs that are located in previous scheduling windows. It only optimizes or adjusts ETAs that are located in the active scheduling window or in future windows. This technique ensures the schedule stability for the immediate future while it retains schedule flexibility for the longer term.

D. Induced Constraints

To ensure sufficient separations, locked ETAs of a given flight are delivered to the static algorithm as separation constraints. Other constraints contained in the MILP formulation also remain valid. Furthermore, dynamic scheduling over multiple windows can potentially result in separation violations between flights located in adjacent windows because of the sequential nature. To eliminate any potential cross-window conflict, the ETAs of a sufficient number of flights toward the end of the preceding window are also included in the scheduling optimizations during the active window as separation constraints. The static algorithm applied over the active window can then sufficiently separate flights in adjacent windows.

IV. Simulation Evaluations

A numerical simulation environment is developed to evaluate the performances of various dynamic scheduling strategies. All simulation programs are coded in Java. Lp_Solve,[§] an open-source MILP solver, is used for the static runway scheduling.

Figures 1 and 2 show the airborne routes and surface routes used in this work. It is assumed that the arriving traffic streams enter the airport through three fixes: LENDY, CAMRN, and ROBER. Flights from ROBER land on runway 31R by default, whereas flights from LENDY and CAMRN use runway 31L instead. All departures takeoff from runway 31L and exit the terminal airspace through the north gate, east gate, west gate, and ocean gate.

Three aircraft classes are considered in the simulation studies: large, heavy, and Boeing 757. Different separation requirements can apply to the same aircraft pair, depending on if they are on the ground or in the air. The airborne and surface separation requirements are listed in Tables 3 and 4, respectively. All separation requirements are expressed in time.

The computational time and the overall delay are used as performance criteria to evaluate different dynamic strategies. Equation (1) shows the overall delay of a dynamic scheduling strategy. It contains the total gate delay for arrivals plus the total fix delay for departures. Besides, both the average and the maximum computational time are considered. The average computational time of a dynamic strategy is obtained by taking the average of computational times used in the series of scheduling windows. By contrast, the maximum computational time represents the largest computational time on a single scheduling window over all the scheduling windows. When the maximum computational time bound is reached at a particular window, the solution process is terminated and the best feasible solutions up to this time are reported.

In this paper, a dynamic strategy that produces a smaller overall delay is regarded as more effective. It is important to recognize that different optimization criteria define different notions of being effective. Furthermore, many practical considerations may not be easily expressed mathematically through optimization criteria or constraints. While optimal solutions provide useful references for practical applications, they may not necessarily represent best solutions in the real world.

Solution robustness to data uncertainties is also a desirable property in practical operations. Indeed, ETAs delivered to the scheduling process likely contain errors. In this regard, sequential dynamic strategies are structurally robust because they use updated information over each scheduling window. Designs of dynamic strategies for achieving a desired level of robustness shall be considered in the future.

[§]Data available at http://lpsolve.sourceforge.net/ [retrieved 21 May 2011].

Table 3 Airborne and runway separations (seconds)

Leader\follower	Large	757	Heavy
Large	50	50	50
757	85	65	65
Heavy	85	85	65

Table 4 Surface separations (seconds)

Leader\follower	Large	757	Heavy
Large	30	30	30
757	30	30	30
Heavy	30	30	30

V. Results on Scheduling Window Sizes and Overlaps

Dynamic strategies with different window sizes and overlap sizes are first tested. Dynamic strategy parameters are summarized in Table 2. The two general static scheduling algorithms developed by Chen et al. [33], i.e., the fix-route and the open-route algorithms, are used as the underlying static solution algorithms. The fix-route algorithm produces optimal arrival times and flight sequences only, whereas the open-route algorithm also produces optimal route assignments, in addition to optimal arrival times and flight sequences.

Chen et al. [33] indicate that using the static multiple-point scheduling scheme, a planning horizon longer than 1 h can no longer be solved in a few minutes under heavy traffic. By contrast, the dynamic strategy of this paper can process traffic over a much longer planning horizon using the rolling window strategies. While exact computational times can vary with implementations of the algorithms and computers, the general idea remains true. The remainder of this section presents performances of various rolling window strategies in processing traffic over a 2 h horizon.

The first traffic scenario in Table 5 is used to test dynamic strategies with scheduling windows ranging from 5 to 15 min. This scenario contains a 2 h traffic and represents initially imbalanced traffic on runways 31L and 31R. Three representative overlap sizes of

Table 5 Traffic demands and distributions

Scenario	Hour	Arrivals	Departures	Rwy 31L[a]	Rwy 31R[a]
Imbalanced traffic	1	42	14	46	10
	2	49	16	54	11
Fairly balanced traffic	1	42	14	27	29
	2	49	16	37	28

[a]Rwy denotes runway.

0, 25, and 50% are tested. A 90 s bound on the maximum computational time is imposed on each scheduling window.

A. Dynamic Strategies with Nonoverlapping Windows

Compared with the use of overlapping windows, the use of nonoverlapping windows over a given planning horizon intuitively requires fewer computations to process the same amount of traffic. A larger advance time provides a longer computational time for obtaining solutions. On the other hand, each of the nonoverlapping windows contains different traffic, except for flights that are scheduled over multiple windows at the multiple scheduling points. As a result, optimal schedules obtained in previous windows may not be used as initial guesses in subsequent windows.

Figures 6 and 7 show the computational times for nonoverlapping windows from 5 to 15 min. Under the same traffic conditions and a given static algorithm, a smaller scheduling window contains fewer flights and runs faster in most cases. For example, the average computational time with the 5 min. window is only one-third of that with the 15 min. window. The maximum computational times in Fig. 7 follow a similar trend. Compared with the average computational time, the maximum computational time increases more drastically as the windows become larger. With either static algorithm, the 90 s bound starts to limits the maximum computational time as the window size exceeds 10 min.

The two static algorithms demonstrate different levels of efficiency. With a given window size, the fix-route algorithm is always faster than the open-route algorithm, as shown in Figs. 6 and 7. This is consistent with results in the previous studies of static runway scheduling [33]. As the window size varies, the open-route algorithm reaches the 90 s bound before the fix-route algorithm does (Fig. 7). In these tests, the average computational time using the fix-route algorithm is always 6–8 s less than that using the open-route algorithm. Trends of the maximum computational time are similar before reaching the 90 s bound.

Before the maximum computational times reach the 90 s bound, the overall delay and the window size vary inversely, as shown in Fig. 8. A dynamic strategy with larger window sizes produces less delays. As soon as the bound on the maximum computational time is reached, on the other hand, this inverse relation between the overall delay and the window size no longer holds. Instead, further increase in the window size produces more delays. This is caused by the suboptimal nature of solutions when computations are terminated at 90 s. Computational loads on the static algorithms grow as the scheduling window size increases further, and these increased computational loads degrade the solution optimality.

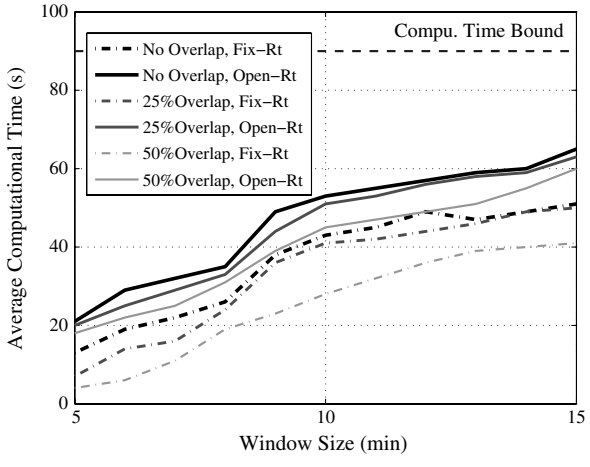

Fig. 6 Average computational times with various dynamic strategies.

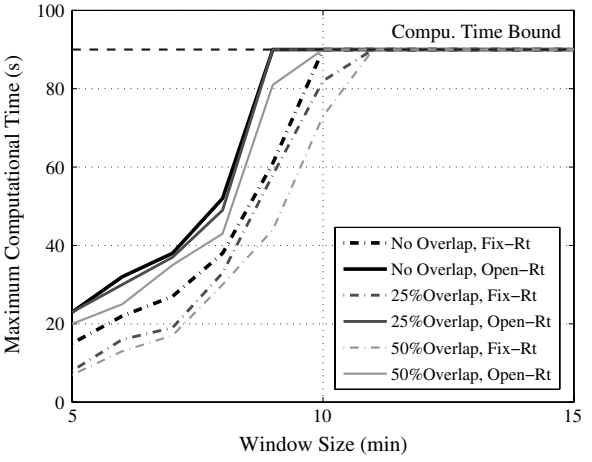

Fig. 7 Maximum computational times with various dynamic strategies.

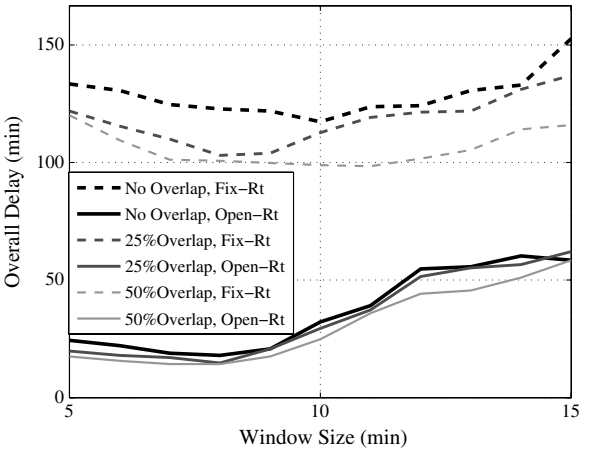

Fig. 8 Overall flight delays with various dynamic strategies.

B. Dynamic Strategies with Overlapping Windows

Intuitively, overlaps between neighboring scheduling windows may potentially speed up the computations, especially for large windows. This is because for two adjacent windows, schedules of the overlapping traffic that have been optimized in the previous window serve as good initial guesses for fine-tuning in the subsequent window. This fine-tuning can be completed in a shorter time.

As shown in Figs. 6 and 7, a 25% overlap between neighboring windows can reduce both the average and the maximum computational times compared with nonoverlapping windows. Specifically, the 25% overlap can reduce the average computational time by 2–5 s

Table 6 Runway scheduling options to be tested

Option	Runway assignment rule	Sequencing rule
FCFS	Default runway assignments	FCFS sequences
Sequencing	Default runway assignments	Optimal sequences
Routing	Optimal runway assignments	FCFS sequences
Comprehensive scheduling	Optimal runway assignments	Optimal sequences

using the open-route algorithm. The reduction is even larger with the fix-route algorithm. It also permits the use of a larger window size before the fix-route algorithm reaches the 90 s bound.

In terms of the overall performance, dynamic strategies with 25% overlaps follow somewhat similar trends as those with non-overlapping windows. Before the maximum computational time reaches the 90 s bound, an inverse relation again exists between the overall delay and the window size, as shown in Fig. 8. The overall delays with the fix-route algorithm can be reduced substantially if windows are less than 10 min. This reduction, however, diminishes as the window size increases further, as the bound on the maximum computational time starts to become constraining. On the other hand, the 25% overlap does not provide the open-route algorithm with equal reductions in overall delays. Figure 8 shows that the overall delay would no longer decrease once the maximum computational time reaches the 90 s bound. It even starts rising if the scheduling window size increases further.

A reasonable increase in the overlap can improve both the efficiency and the overall performance. As shown in Fig. 6, a 50% overlap between neighboring windows can substantially reduce the average computational times of both static algorithms, especially when window sizes are large. Besides, Figs. 6 and 7 indicate that the fix-route algorithm results in more reductions in computational times than the open-route algorithm. The 50% overlap also produces reductions in the overall delay, as shown in Fig. 8. In particular, the fix-route algorithm results in greater reductions than the open-route algorithm, especially when window sizes are greater than 10 min. For overlap sizes within a certain range, the fix-route algorithm is faster than the open-route algorithm, but the open-route algorithm produces better scheduling performances than the fix-route algorithm. Overly large overlaps result in redundant computations that may adversely affect the performances of a dynamic strategy.

Overall, the smallest overall delay occurs with a 8 min. window size, a 50% overlap, and the use of the open-route algorithm. Reducing the overlap to 25% results in a slight increase in the overall delay. On the other hand, the fastest dynamic strategy employs a 5 min. scheduling window, a 50% overlap, and the fix-route algorithm.

VI. Benefits of Optimal Sequencing and Runway Assignments

To evaluate the benefits of optimal sequencing and runway assignments, four levels of scheduling optimization options are constructed in Table 6. A basic first-come–first-served (FCFS) scheduling principle is introduced for comparison purposes. In the FCFS option, the original runway assignments are maintained and flight sequences are determined on a FCFS basis at merge points. By contrast, the sequencing option allows aircraft to switch their sequences at merge points while maintaining their original flight runway assignments. This is achieved by the fix-route algorithm. Next, the routing option generates optimal runway and route assignments for individual aircraft while maintaining their original flight sequences at the merge points on a FCFS basis. This option is essentially a tradeoff between the fix-route and the open-route algorithms. Finally, the comprehensive scheduling option permits both flight sequence changes and runway assignment adjustments. This option is equivalent to the open-route algorithm and provides the maximum freedom in runway scheduling. The comprehensive scheduling option is expected to offer the best performance.

A representative dynamic strategy is assumed, which uses a series of 10-min.-long nonoverlapping scheduling windows. This dynamic

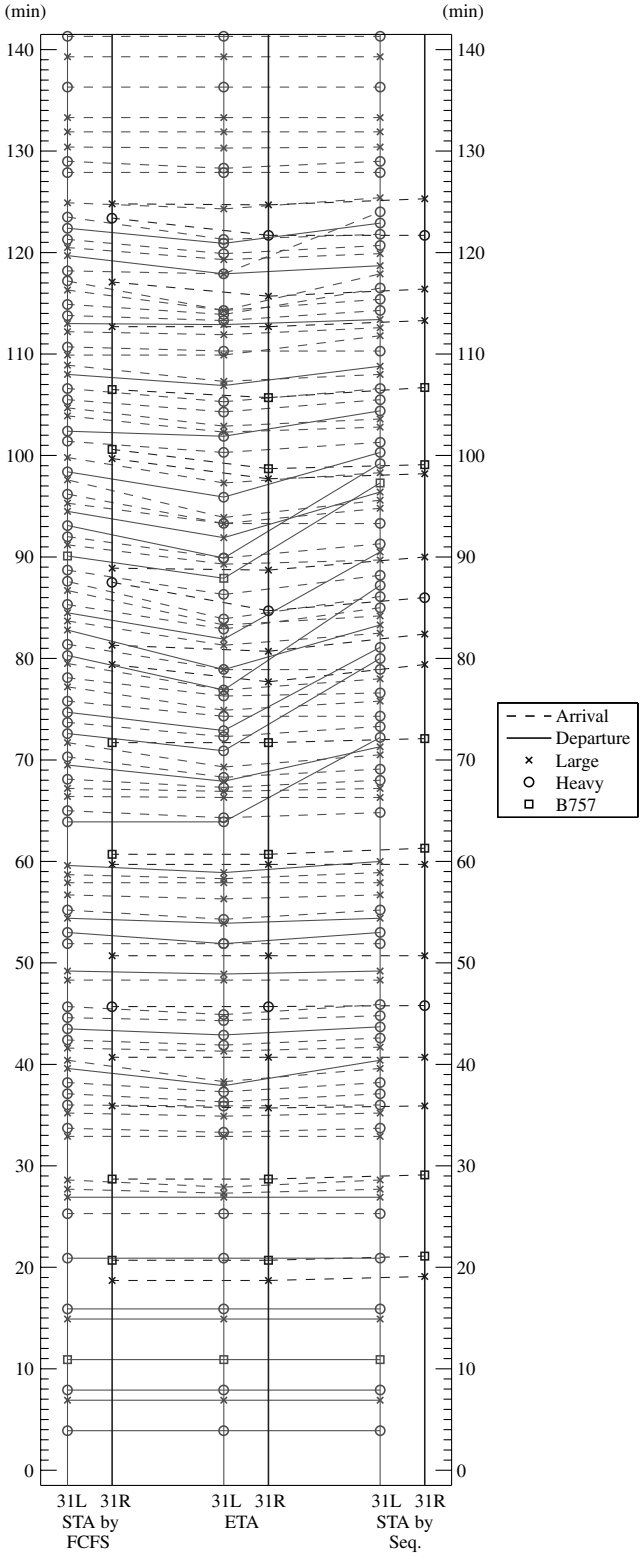

Fig. 9 Sequencing option with imbalanced traffic.

strategy provides a balance between computational speed and scheduling effectiveness for the two traffic scenarios listed in Table 5, which emulate different operational environments. In fact, both scenarios correspond to an arrival rush in the John F. Kennedy International Airport (JFK) but represent different initial traffic loads on runways 31L and 31R. Specifically, the first scenario represents an imbalanced initial traffic on runways 31L and 31R. It is used to test the ability of the dynamic strategy in balancing the traffic loads on the two runways. By contrast, the second scenario contains fairly balanced initial traffic loads on runways 31L and 31R. It is used as a comparison case to study the increase of computational efforts when initial traffic loads are imbalanced, as in the first traffic scenario.

Sequence plots as shown in Figs. 9–17 are devised to display the solutions. Each sequence plot shows individual flights' runway ETAs, STAs, and flight sequences on the runways. In addition, a sequence plot also expresses individual flights' runway delays and their sequence adjustments. In each sequence plot, the two vertical lines in the middle represent runways 31L and 31R. Each mark on

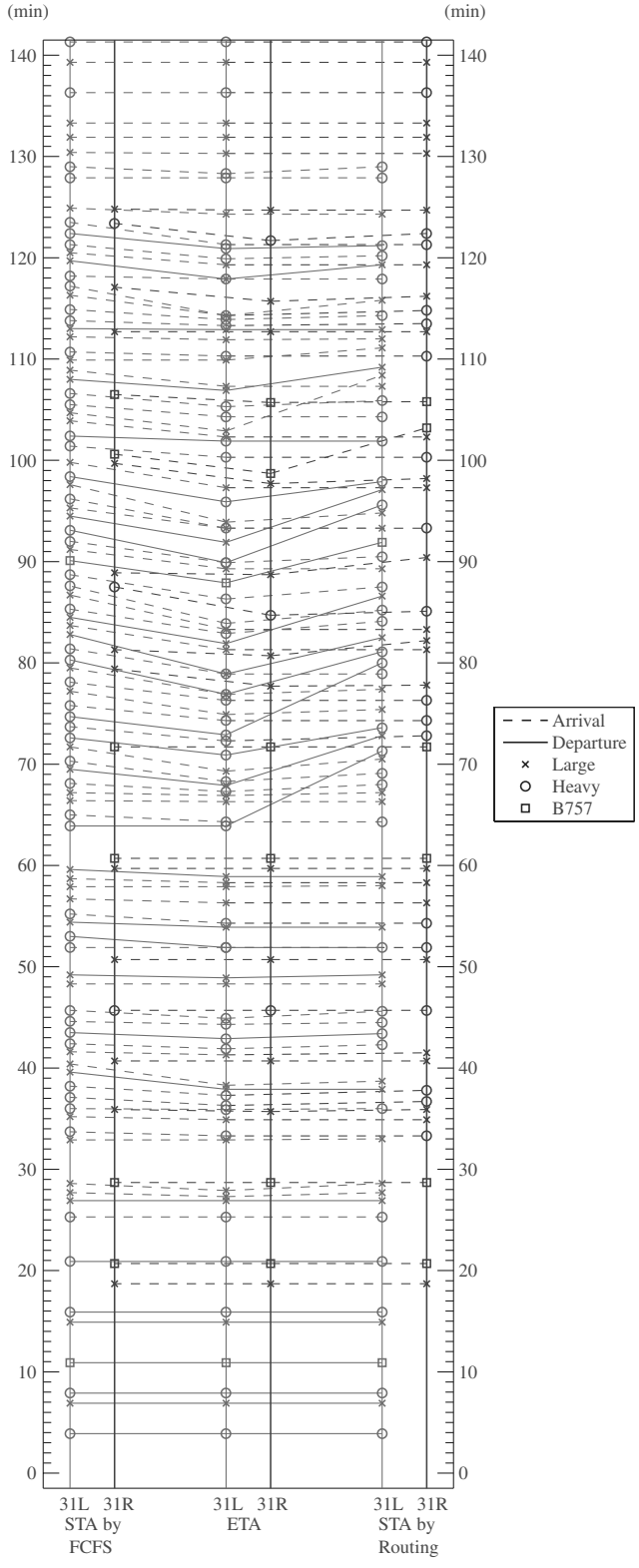

Fig. 10 Routing option with imbalanced traffic.

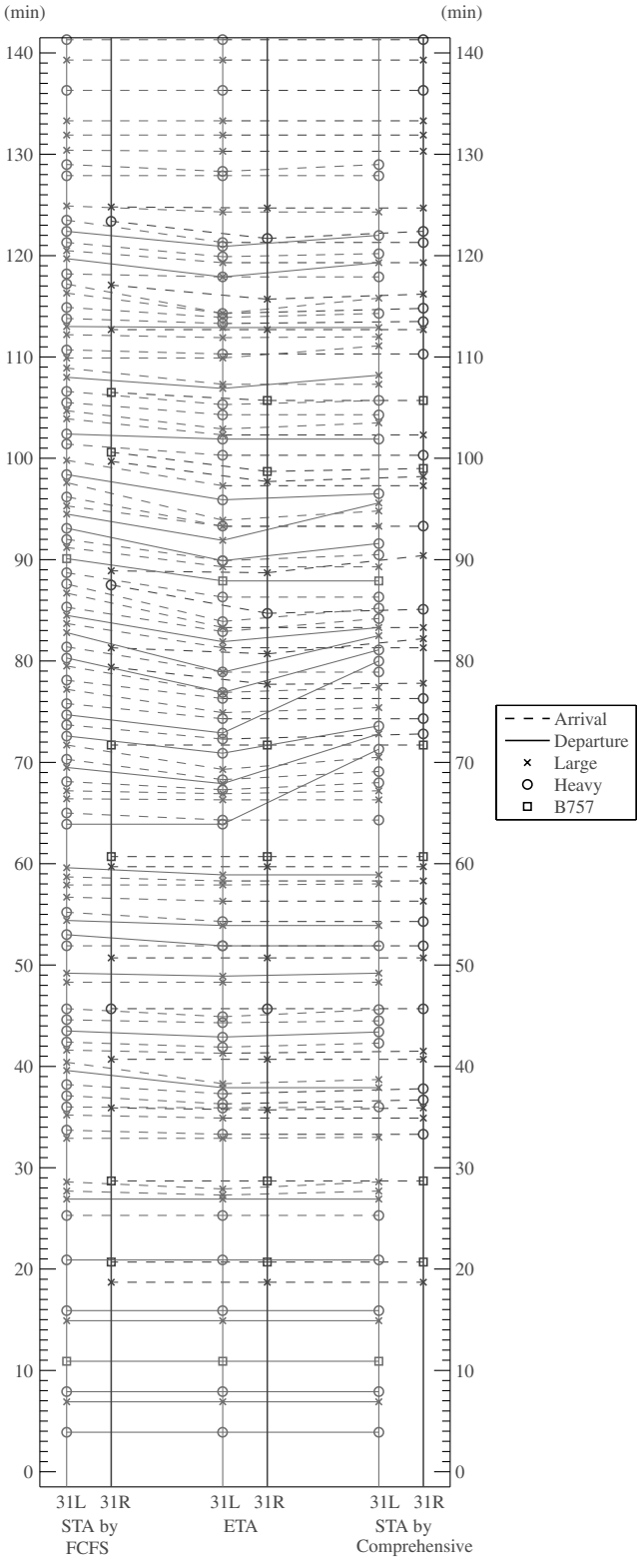

Fig. 11 Comprehensive scheduling option with imbalanced traffic.

these lines represents the arrival/departure time of a particular aircraft that either lands on or takes off from the corresponding runway. Different types of marks stand for different types of aircraft.

Besides the two vertical lines in the middle, there are two lines on the right and two on the left. These two sets of vertical lines display the STAs produced by two different runway scheduling algorithms, such as the FCFS method and dynamic scheduling with sequencing, as shown in Fig. 9. Similarly, the marks on each vertical line display the STAs and new sequences generated by the dynamic scheduling process. By connecting a flight's ETA and its STA in the figure, it can be clearly seen if the aircraft is delayed or not in the scheduling process. By completing all lines connecting ETAs and their corresponding STAs, one can tell if the flight sequences or runway assignments have been changed. In sequence plots, runway STAs (runway arrival/departure times) are expressed in relative times instead of absolute times. For example, if the scheduling horizon begins at 1800 hrs, the 0 min. on the plot corresponds to 1800 hrs, and 60 min. corresponds to 1900 hrs.

A. Results for Imbalanced Traffic

This scenario simulates an initially imbalanced traffic on runways 31L and 31R (Table 5). As shown in Figs. 9 and 19, neither the FCFS option nor the sequencing option improves the traffic imbalance on the two runways. Also, no arrival traffic on runway 31L is directed to runway 31R, which has the potential to serve more flights. In attempting to relieve the congestion on runway 31L, the dynamic strategy can only adjust the flight sequences on both runways. As a result, several departure flights are held on the ground a little longer so that runway 31L can serve arrivals first.

For example, departures FD22, FD23, and FD24 are held on the ground in order to accept some arriving flights at the same time period (Fig. 12). The three aforementioned departing flights are delayed and then automatically inserted into vacancies in the queue on runway 31L. Furthermore, FD22 and FD24 are also swapped in the queue. Because FD22 is a B-757, less separation is required when it departs behind the large aircraft FD24 than the other way around. This swap reduces the runway occupancy

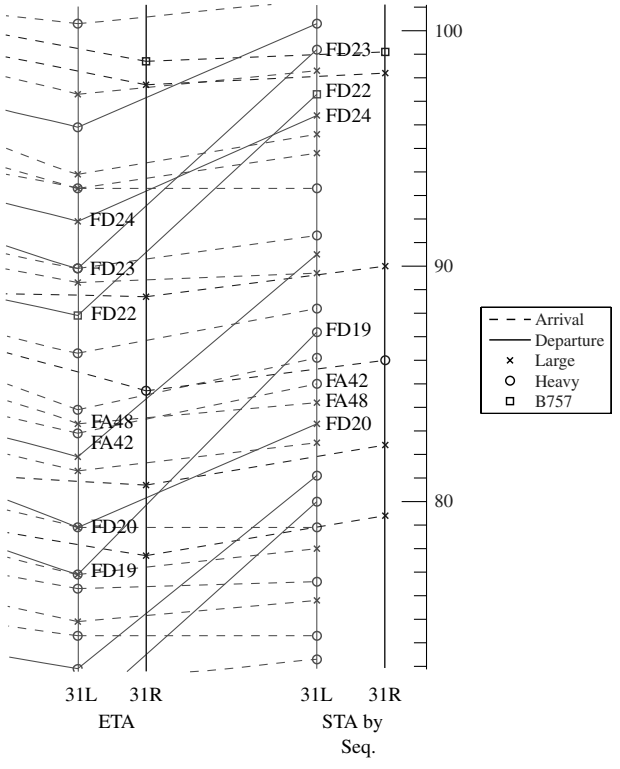

Fig. 12 Sequencing option with imbalanced traffic (zoom-in).

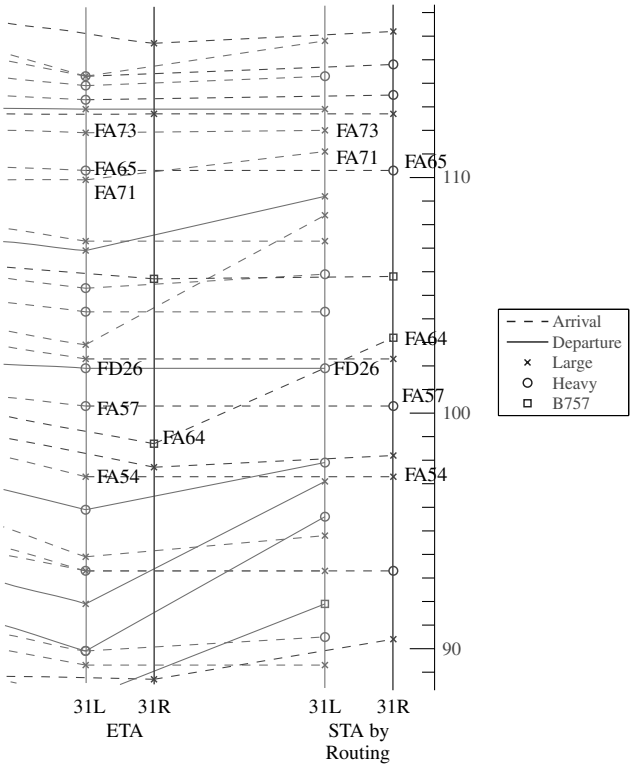

Fig. 13 Routing option with imbalanced traffic (zoom-in).

time for this aircraft pair. In another example, FD20 is delayed by about 3 min. and then inserted into the vacancy before FA48. This new arrangement reduces the delays imposed on FA42, FA48, and FAD20. Meanwhile, FA42 and FA48 are also swapped for a smaller separation.

These results indicate that the dynamic strategy with the sequencing option is able to adjust the flight sequences intelligently. It can properly insert flights into the vacancies of the queues to incur less delays on other flights. The overall performance is improved by using sequence adjustments as shown in Fig. 18. Compared with the FCFS option, the sequencing option can reduce the overall delay from over 110 min. to around 75 min.

Next, the routing option is tested to examine the benefits of optimizing runway assignments. In this simulation setup, flights originally planned to land on runway 31L are permitted to land on runway 31R. On the other hand, departures retain their default

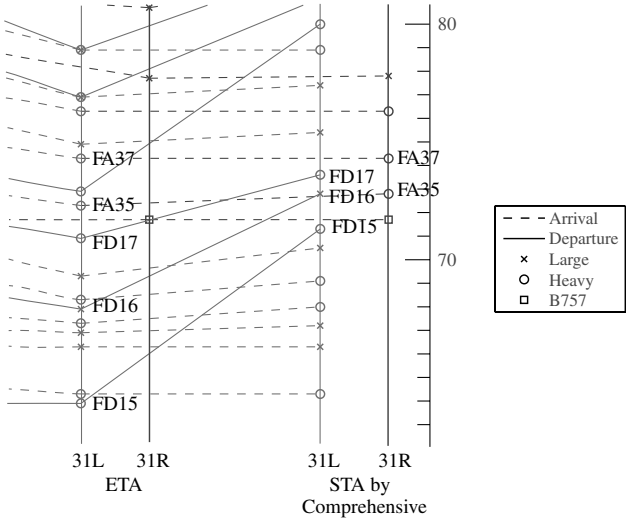

Fig. 14 Comprehensive scheduling option with imbalanced traffic (zoom-in).

runway assignments. Since runway 31L has a heavier traffic load due to the mixed operations, the dynamic strategy is expected to intelligently balance the traffic loads on runways 31L and 31R to reduce the overall flight delay. As shown in Fig. 10, sequences on both runways are mostly maintained, with some overflow traffic directed to runway 31R. Figure 19 shows that runway 31R now serves 28 more flights than in the original plan. An average of 14 runway assignment changes per hour is used to reduce the congestion on runway 31L.

Figure 13 provides examples of runway assignment adjustments. Arrival FA65, which is originally right behind FA71 with insufficient separation, is directed to runway 31R with no delay imposed. Also, FA73 can land right after FA71 without a significant delay. Meanwhile, FA57 and FA54 are also directed to runway 31R to avoid delays. FD26 takes off on time also, benefiting from the new runway assignment of FA57.

In terms of the overall flight delay, the routing option is more efficient than the sequencing option, as shown in Fig. 18. The overall

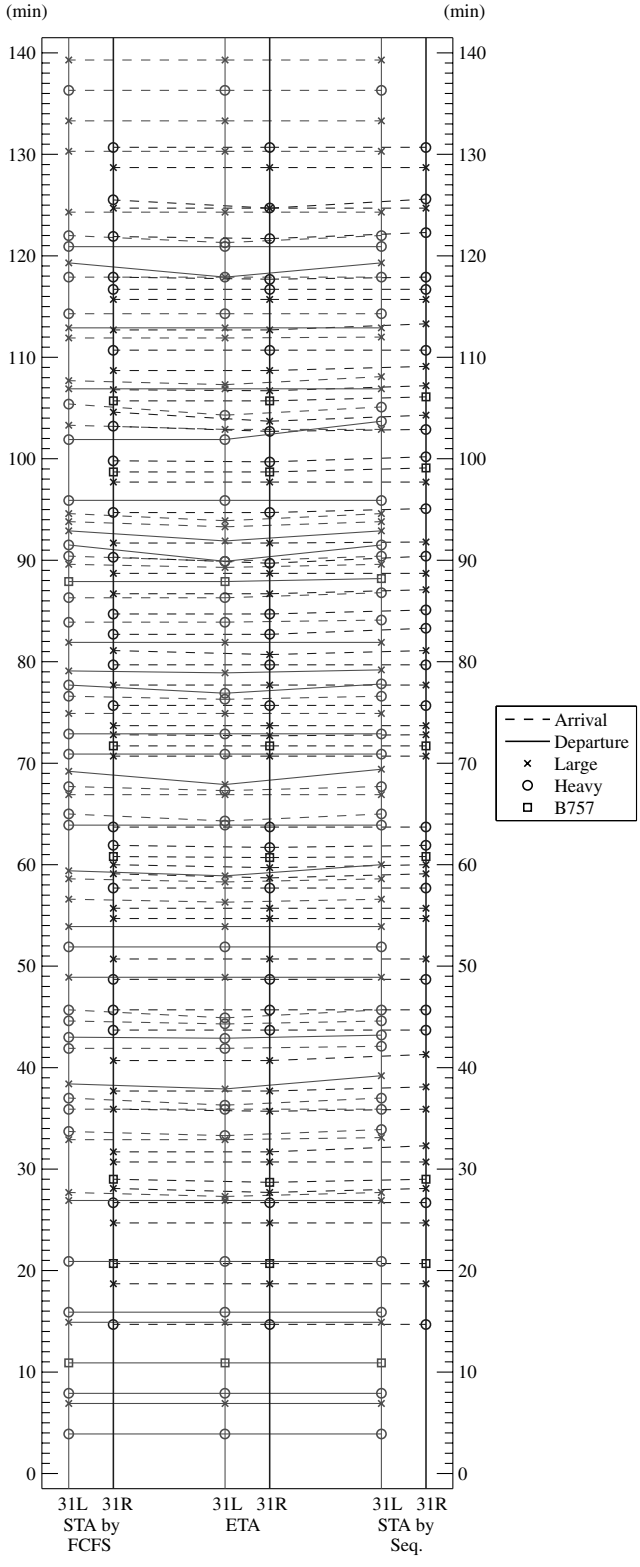

Fig. 15 Sequencing option with fairly balanced traffic.

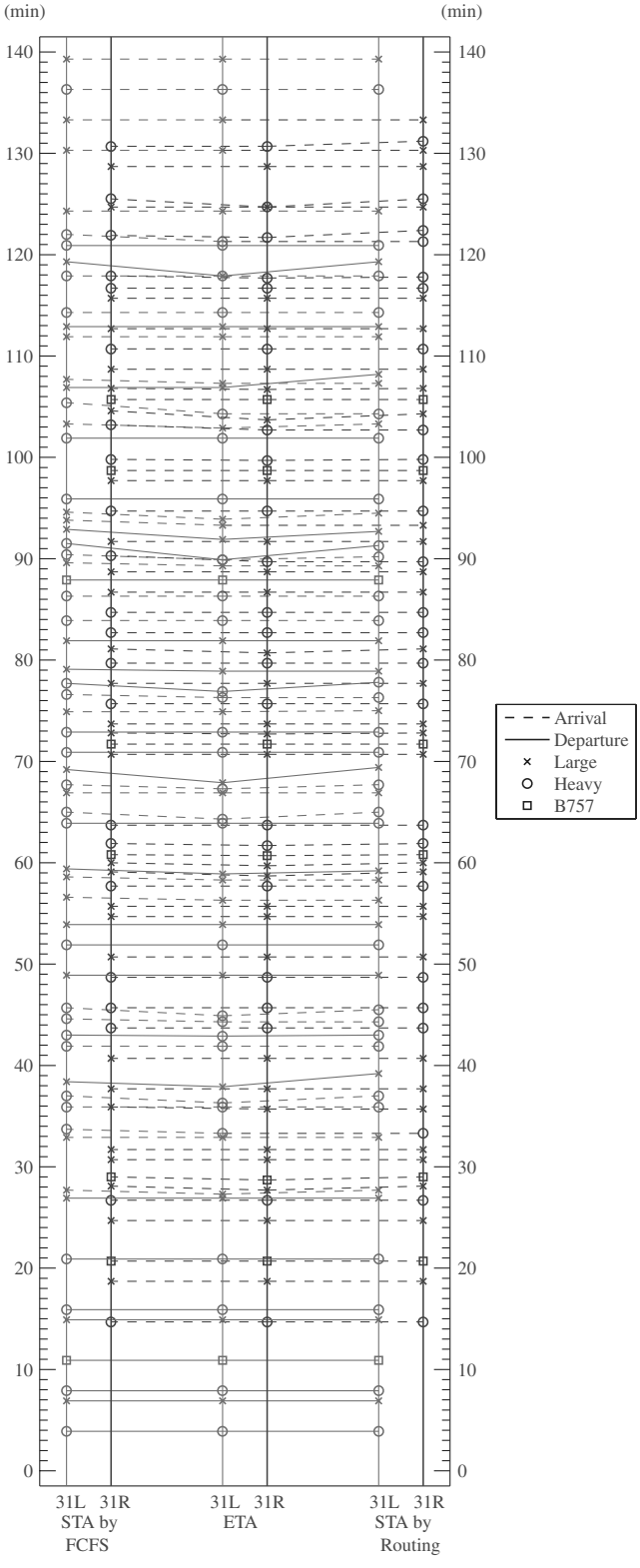

Fig. 16 Routing option with fairly balanced traffic.

delay with the routing option is only around 60 min., which is 15 min. less than the sequencing option and 40 min. less than the FCFS option. As illustrated in Fig. 19, a total of 28 flights are directed from runway 31L to runway 31R, which produces a ratio of 72:49 between the number of flights on runways 31L and 31R. These results suggest that the routing option is more efficient than both the FCFS option and the sequencing option.

Finally, the comprehensive scheduling option permits adjustments in both flight sequences and runway assignments. As shown in Fig. 11, necessary sequence adjustments are conducted along with traffic balancing. Many arrivals are directed to runway 31R to release the congestion on runway 31L, and new sequences are produced. Similar to the routing option, Figs. 11 and 19 indicate a total of 28 runway assignment changes. Figure 18 indicates that the congestion on runway 31L has been greatly released and the overall flight delay is further reduced to 36 min., which is the least among all the options.

Figure 14 offers examples of simultaneous traffic balancing and sequencing. For instance, FA37 is directed to 31R to avoid delays due to the separation requirement between FA37 and FD17. Besides, FD15, FD16, and FD17 are held on the ground until vacancies in the arrival queue appear. The heavy aircraft FD15 is sequenced behind a large arrival aircraft so that a smaller separation between them applies. The new runway assignment for FA35 not only resolves the conflicts among FD15, FD16, and FD17 but also imposes the minimum delays on these flights.

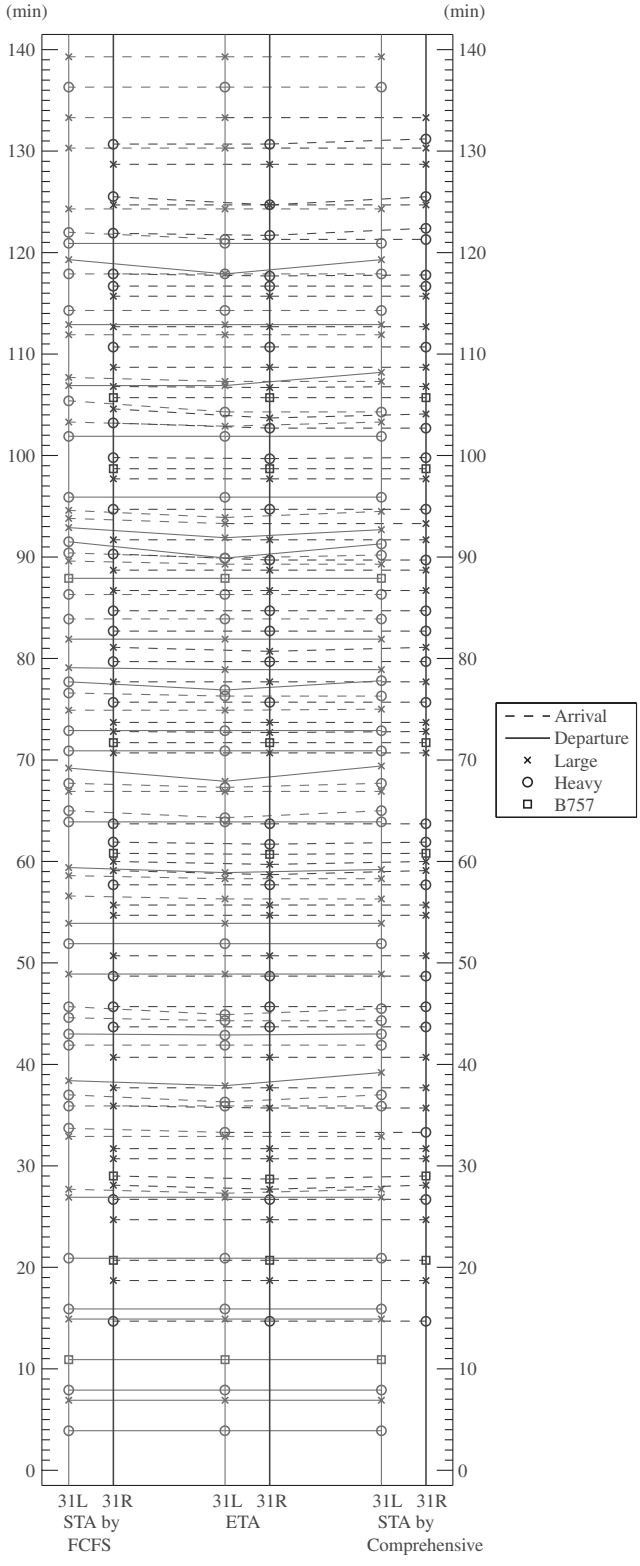

Fig. 17 Comprehensive scheduling option with fairly balanced traffic.

B. Results for Fairly Balanced Traffic

In this scenario, it is assumed that the traffic loads on runways 31L and 31R are fairly balanced. The dynamic strategy mainly focuses on sequencing and separation adjustments.

Again, the sequencing option only permits sequence adjustments. No traffic is directed from runways 31L to 31R. As shown in Figs. 15 and 19, neither the FCFS option nor the sequencing option has altered the original runway assignments. In this case, only a minimum amount of arrival–departure swaps are produced and the original sequences are basically maintained on both runways. A few departures are held on the ground so that the runway can serve arrivals first. These results suggest that, although the initial traffic demand is not challenging, the dynamic strategy can still refine the original sequence to produce less delays. Because of the initially balanced traffic loads on runways 31L and 31R, the overall delay in Fig. 18 is quite small compared with the previous scenario with initially imbalanced traffic. Even under such a circumstance, the sequencing option still outperforms the FCFS option.

Similar to the initially imbalanced traffic scenario, the routing option further reduces the overall delay somewhat, as shown in Figs. 16 and 18. Both the FCFS option and the routing option maintain most flight sequences. Meanwhile, a total of six runway assignment changes are made, which is smaller compared with the first scenario with imbalanced initial traffic between the two runways.

Finally, the comprehensive scheduling option still offers the largest freedom and achieves the minimum overall delay in this scenario. It seems that the routing option and the comprehensive scheduling option eventually assign the same runway traffic loads, as shown in Figs. 17 and 19. However, the comprehensive scheduling option produces a better sequencing plan with a smaller overall delay (Fig. 18). In all options, the overall delays produced in this scenario are much smaller than those in the first scenario.

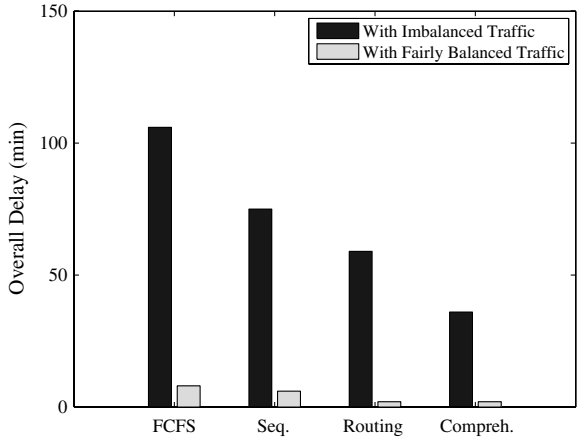

Fig. 18 Overall delays with various scheduling options.

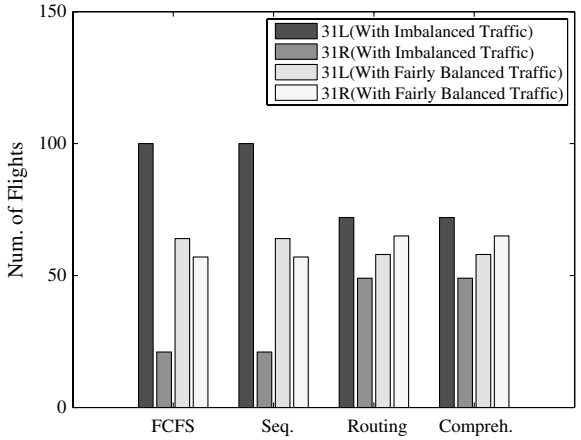

Fig. 19 Optimized runway traffic loads with various scheduling options.

VII. Conclusions

This paper presents dynamic strategies for real-time integrated scheduling and runway assignment of terminal area traffic including both arrivals and departures. A dynamic strategy divides the planning horizon into a series of smaller scheduling windows and applies an appropriate static scheduling algorithm to obtain solutions for each window sequentially. In this paper, a multiple-point integrated scheduling scheme and its static MILP formulation are briefly reviewed. Problems with the direct application of static scheduling algorithms are identified that include the inability to schedule continual traffic streams, lack of computational time guarantee, and inability to take advantage of updated traffic information. Then, structures and parameters of general dynamic scheduling strategies are proposed. These parameters include scheduling window sizes as well as overlaps between adjacent windows. Furthermore, dynamic strategies can be designed to optimize times of arrival only, or to include the optimization of sequences at merge points and runway assignments as well, through proper specifications of the route network in the underlying static scheduling algorithms. Besides necessary constraints for physical and operational feasibility, the need for additional constraints between neighboring windows in order to ensure sufficient separations is discussed. In using extensive numerical simulations to evaluate the proposed dynamic strategies, a single airport model with multiple runways is assumed. The JFK airport is used as the example in the simulations.

Simulation results indicate that dynamic strategy parameters can be selected to enhance scheduling performances, to reduce computational times, or to achieve a proper balance between the two. Using the same parameters, a dynamic strategy with overlapping windows is faster than that with nonoverlapping windows. A larger degree of freedom in scheduling strategies generally results in better scheduling performances at the expense of longer computational times. Specifically, dynamic strategies with the ability to adjust both sequences and runway assignments provide better scheduling performances than those that can only adjust sequences or runway assignments. When only one choice is available, the ability to adjust runway assignments provides more scheduling benefits than the ability to optimize sequences, especially under initially imbalanced traffic among the runways. Even when both sequences at merge points and runway assignments are fixed, dynamic strategies can still be used to optimize STAs at multiple points and perform better than a simple FCFS strategy.

Acknowledgments

This research was supported in part by NASA under the NASA Research Announcement contract "System Oriented Runway Management" to Mosaic ATM, Inc., with the University of Minnesota as a subcontractor. The authors thank Steve Atkins and Chris Provan for many helpful discussions.

References

[1] "Next Generation Air Transportation System (NextGen): Flight Prioritization Deep Dive: Final Report," Joint Planning and Development Office, Washington, D.C., Jan. 2011, http://www.jpdo.gov/library/20110113_FP_Report_Final_v3.pdf [retrieved 2011].

[2] Psaraftis, H. N., "A Dynamic Programming Approach to the Aircraft Sequencing Problem," Massachusetts Inst. of Technology Flight Transportation Lab. Rept. FTL R78-4, Cambridge, MA, 1978.

[3] Abela, J., Abramson, D., Krishnamoorthy, M., De Silva, A., and Mills, G., "Computing Optimal Schedules for Landing Aircraft," *12th National Conference of the Australian Society for Operations Research*, Adelaide, Australia, July 1993, pp. 71–90.

[4] Dear, R. G., and Sherif, Y. S., "An Algorithm for Computer Assisted Sequencing and Scheduling of Terminal Area Operation," *Transportation Research*, Vol. 25, 1991, pp. 129–139. doi:10.1016/0191-2607(91)90132-A

[5] Ernst, A. T., Krishnamoorthy, M., and Storer, R. H., "Heuristic and Exact Algorithms for Scheduling Aircraft Landings," *Networks*, Vol. 34, 1999, pp. 229–241. doi:10.1002/(SICI)1097-0037(199910)34:3<229::AID-NET8>3.0.CO;2-W

[6] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D., "Scheduling Aircraft Landings: The Static Case," *Transportation Science*, Vol. 34, No. 2, May 2000, pp. 180–197. doi:10.1287/trsc.34.2.180.12302

[7] Beasley, J. E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D., "Displacement Problem and Dynamically Scheduling Aircraft Landings," *Journal of the Operational Research Society*, Vol. 55, 2004, pp. 54–64. doi:10.1057/palgrave.jors.2601650

[8] Kupfer, M., "Scheduling Aircraft Landings to Closely Spaced Parallel Runways," *8th USA/Europe Air Traffic Management Research and Development Seminar*, 2009.

[9] Chandran, B., and Balakrishnan, H., "A Dynamic Programming Algorithm for Robust Runway Scheduling," *Proceedings of the American Control Conference*, New York, July 2007, pp. 1161–1166.

[10] Lee, H., and Balakrishnan, H., "Fuel Cost, Delay and Throughput Tradeoffs in Runway Scheduling," *Proceedings of the American Control Conference*, Seattle, WA, American Automatic Control Council, Dayton, OH, June 2008, pp. 2449–2454.

[11] Rathinam, S., Wood, Z., Sridhar, B., and Jung, Y. C., "A Generalized Dynamic Programming Approach for a Departure Scheduling Problem," AIAA Guidance, Navigation, And Control Conference, Chicago, IL, AIAA Paper 2009-6250, 2009.

[12] Atkin, J. A. D., Burke, E. K., Greenwood, J., and Reeson, D., "The Effect Of The Planning Horizon And The Freezing Time On Take-off Sequencing," *Proceedings of 2nd International Conference on Research in Air Transportation (ICRAT2006)*, Belgrade, Serbia, June 2006.

[13] Gupta, G., Malik, W., and Jung, Y., "A Mixed Integer Linear Program for the Airport Departure Scheduling Problem," 9th AIAA Aviation Technoology, Integration, and Operations Conference (ATIO), Hilton Head, SC, AIAA Paper 2010-7692, Sept. 2009.

[14] Bianco, L., Dell'Olmo, P., and Giordani, S., "Scheduling Models and Algorithms for TMA Traffic Management," *Modeling and Simulation in Air Traffic Management*, edited by L. Bianco, P. Dell'Olmo, and A. R. Odoni, Springer–Verlag, New York, 1997, pp. 139–167.

[15] Bianco, L., Dell'Olmo, P., and Giordani, S., "Dynamic Algorithms for TMA Traffic Management," *Proceedings of the 8th IFAC/IFIP/IFORS Symposium on Transportation Systems*, edited by M. Papageorgiou, and A. Pouliezos, Chania, Greece, Pergamon–Elsevier Science, New York, 1998, pp. 41–46.

[16] Mohleji, S. C., and Jacobs, F., "Airspace Design and Arrival/Departure Planning for Brussels National Airport," *Proceedings of 3rd USA/Europe Air Traffic Management R&D Seminar*, Napoli, Italy, June 2000.

[17] Smeltink, J., Soomer, M., DeWaal, P., and Van Der Mei, R., "An Optimisation Model for Airport Taxi Scheduling," *30th Conference on the Mathematics of Operations Research*, Lunteren, The Netherlands, Jan. 2005.

[18] Marin, A., "Airport Management: Taxi Planning," *Annals of Operations Research*, Vol. 143, 2006, pp. 191–202. doi:10.1007/s10479-006-7381-2

[19] Roling, P., and Visser, H., "Optimal Airport Surface Traffic Planning Using Mixed-Integer Linear Programming," *International Journal of Aerospace Engineering*, Vol. 2008, 2008, Paper 732828. doi:10.1155/2008/732828

[20] Rathinam, S., Montoya, J., and Jung, Y., "An Optimization Model for Reducing Aircraft Taxi Times at the Dallas Fort Worth International Airport," *26th International Congress of the Aeronautical Sciences*, 2008.

[21] Keith, G., Richards, A., and Sharma, S., "Optimization of Taxiway Routing and Runway Scheduling," AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, HI, AIAA Paper 2008-6827, Aug. 2008.

[22] Cheng, Y., "A Knowledge-Based Airport Gate Assignment System Integrated with Mathematical Programming," *Computers and Industrial Engineering*, Vol. 32, No. 4, 1997, pp. 837–852. doi:10.1016/S0360-8352(97)00001-6

[23] Bolat, A., "Assigning Arriving Flights at an Airport to the Available Gates," *Journal of the Operational Research Society*, Vol. 50, No. 1, 1999, pp. 23–34. doi:10.1057/palgrave.jors.2600655

[24] Kim, S. H., Feron, E., and Clarke, J., "Assigning Gates by Resolving Physical Conflicts," AIAA Guidance, Navigation, and Control Conference, Chicago, IL, AIAA Paper 2009-5648, 2009.

[25] Capozzi, B., Atkins, S., and Choi, S., "Towards Optimal Routing and Scheduling of Metroplex Operations," 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), Hilton Head, SC, AIAA Paper 2009-7037, 2009.

[26] Garcia, J., "MAESTRO: A Metering and Spacing Tool," *Proceedings of the 1990 American Control Conference*, San Diego, CA, American Automatic Control Council, Dayton, OH, 1990, pp. 501–507.

[27] Volckers, U., "Arrival Planning and Sequencing with COMPAS-OP at the Frankfurt ATC-Center," *Proceedings of the 1990 American Control Conference*, San Diego, CA, American Automatic Control Council, Dayton, OH, 1990, pp. 496–501.

[28] Erzberger, H., Davis, T. J., and Green, S. M., "Design of Center-TRACON Automation System," *AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management*, Berlin, May 1993.

[29] Davis, T. J., Robinson, J.E., III, Isaacson, D. R., den Braven, W., Lee, K. K., and Sanford, B. D., "Operational Test Results of the Final Approach Spacing Tool," *Proceedings of the 8th IFAC Symposium on Transportation Systems*, Chania, Greece, June 1997.

[30] Erzberger, H., "Design Principles and Algorithms for Automated Air Traffic Management," AGARD Lecture Series 200, 1995.

[31] Robinson, J. E., III, Davis, T. J., and Isaacson, D. R., "Fuzzy Reasoning-Based Sequencing of Arrival Aircraft in the Terminal Area," AIAA Guidance, Navigation and Control Conference, New Orleans, LA, AIAA Paper 1997-3542, Aug. 1997.

[32] Wong, G. L., "The Dynamic Planer: the Sequencer, Scheduler, and Runway Allocator for Air Traffic Control Automation," NASA Ames Research Center TM-2000-209586, Moffett Field, CA, 2000.

[33] Chen, H., Zhao, Y. J., and Provan, C., "Multiple-Point Integrated Scheduling of Terminal Area Traffic," *Journal of Aircraft*, Vol. 48, No. 5, 2011, pp. 1646–1657. doi:10.2514/1.55322

[34] Lee, H. W. J., Teo, K. L., and Cai, X. Q., "An Optimal Control Approach to Nonlinear Mixed Integer Programming Problems," *Computers and Mathematics with Applications*, Vol. 36, No. 3, Aug. 1998, pp. 87–105. doi:10.1016/S0898-1221(98)00131-X

[35] Dmitruk, A. V., and Kaganovich, A. M., "The Hybrid Maximum Principle is a Consequence of Pontryagin Maximum Principle," *Systems and Control Letters*, Vol. 57, No. 11, Nov. 2008, pp. 964–970. doi:10.1016/j.sysconle.2008.05.006www.optimization-online.org/DB_FILE/2006/11/1511.pdf [retrieved 2011].