AIAA Guidance, Navigation, and Control Conference
08 - 11 August 2011, Portland, Oregon

AIAA 2011-6380

Runway Scheduling Using Generalized Dynamic Programming

Justin Montoya *
NASA Ames Research Center, Moffett Field CA, 94035

Zachary Wood †
Univ. of Calif., Santa Cruz, Moffett Field, CA 94035

Sivakumar Rathinam ‡
Texas A&M University, College Station, TX 77843

A generalized dynamic programming method for finding a set of pareto optimal solutions for a runway scheduling problem is introduced. The algorithm generates a set of runway flight sequences that are optimal for both runway throughput and delay. Realistic time-based operational constraints are considered, including miles-in-trail separation, runway crossings, and wake vortex separation. The authors also model divergent runway takeoff operations to allow for reduced wake vortex separation. A modeled Dallas/Fort Worth International airport and three baseline heuristics are used to illustrate preliminary benefits of using the generalized dynamic programming method. Simulated traffic levels ranged from 10 aircraft to 30 aircraft with each test case spanning 15 minutes. The optimal solution shows a 40-70 percent decrease in the expected delay per aircraft over the baseline schedulers. Computational results suggest that the algorithm is promising for real-time application with an average computation time of 4.5 seconds. For even faster computation times, two heuristics are developed. As compared to the optimal, the heuristics are within 5% of the expected delay per aircraft and 1% of the expected number of runway operations per hour and can be 1000x faster.

Nomenclature

Q_i	An ordered set of departure aircraft in departure queue i.
G_i	An ordered set of departure aircraft using gate i.
C_i	An ordered set of aircraft that cross the runway at runway crossing queue i.
q_i	The total number of aircraft initially at departure/gate/crossing queue i.
l_i	The number of aircraft remaining in departure/gate/crossing queue i.
m_i	The queue the last departure aircraft uses, which departs to heading i.
a^i_j	Aircraft at departure/crossing/gate queue i in position j from the back.
P_i	The set of aircraft that must not use the runway before aircraft i.
$\alpha(a)$	The earliest time aircraft a can arrive to the runway.
$M(a)$	The weight class of aircraft a.
$fix(a)$	The departure fix of aircraft a.
S	A state.
S_s	A state S with corresponding history sequence s.
$H^{last}_h(S_s)$	The time an aircraft last left to heading h at state S with corresponding history sequence s.
$F^{last}_f(S_s)$	The time an aircraft last left to fix f at state S with corresponding history sequence s.
$C^{last}_c(S_s)$	The time an aircraft last left from crossing c at state S with corresponding history sequence s.
$D^{last}(S_s)$	The time a departure last used the runway at state S with corresponding history sequence s.

*Aerospace Engineer, NASA Ames Research Center,Moffett Field, CA 94035. AIAA Member.
†Software Engineer, University of California, Santa Cruz, Moffett Field, CA 94035. AIAA Member.
‡Assistant Professor, Dept. of Mech. Eng., Texas A & M University, College Station, TX 77843. AIAA Member.

$TIME(S_s)$	The time an aircraft last used the runway at state S with corresponding history sequence s.
$DELAY(S_s)$	The total delay at state S with corresponding history sequence s.
$Sep[x][y]$	The time based standard wake vortices separation between weight class x trailing weight class y.
$R[x][y]$	The time based reduced wake vortices separation between weight class x trailing weight class y.
MIT_f	The time based miles-in-trail separation required between two consecutive departures heading to fix f.
$U(j)$	A binary function equal to $true$ if aircraft j has used the runway and equal to $false$ otherwise.
X	The time based separation between two consecutive runway crossings leaving from the same crossing.
DEP	The amount of time a departure takes to clear the runway.
Sep_c	The amount of time a runway crossing takes to cross the runway at crossing c.
FIX	The total number of departure fixes.
n	The total number of departure headings.
k_1	The number departure queues.
k_2	The number of departure gate queues.
k_3	The number of runway crossing queues.
K	$k_1 + k_2 + k_3$

I. Introduction

The National Airspace System is a complex transportation network with various operational control points, users, policies, and facilitators. While airport systems are small in comparison to the size of the National Airspace System, they represent an important control point for managing aircraft. Runway operations, as one part of the airport system, serve as a major bottleneck for the National Airspace System. For example, the authors in [1] show that take off surface delays, measured as the excess time over the scheduled take-off time, account for over 50% of the National Airspace System delays. Additionally, an analysis using queuing models indicated that the cause of these delays is an imbalance between runway capacity and runway demand [2]. To alleviate this problem, many airports have attempted to expand their capacity by building new runways, taxiways, and gates. Unfortunately, this solution has limited value because many airport systems have exhausted their physical space or are constrained by various environmental regulations. In addition, [3] found that air traffic controllers use simple heuristics for scheduling. When trying to schedule a runway, air traffic controllers are faced with the problem of ordering aircraft to use the runway so that they yield maximum runway efficiency. Since there are a large number of operational considerations including wake vortex separation, aircraft equipage, traffic management initiatives, and runway crossings, the task of safely and efficiently scheduling a runway is difficult in practice. This suggests then, that there exist more efficient solutions for finding aircraft runway schedules than what is currently practiced. In particular, this paper addresses the issue of finding aircraft runway schedules that are optimal for both throughput and delay.

While there are several attempts at finding efficient aircraft runway schedules in the literature, these attempts are either not well-suited for application because of large computation times [4][5][6], do not provide aircraft runway schedules which are optimal for both throughput and delay [4][5][6][7][8][9], or solve simpler variants of a runway scheduling problem [10][8][7]. The authors in [4][5][6], for instance, formulate a runway scheduling problem as a mixed integer linear program. These solutions, however, show poor computational performance. Authors in [4][5][6][7][8][9] attempt to solve the problem by considering only one objective (e.g. either delay or throughput) or explore heuristics that do not guarantee optimal aircraft runway schedules. Lastly, the authors in [7][8][10] solve simpler variants of a runway scheduling problem by considering only departure aircraft, preventing large deviations from a first-come-first-served solution, or ignoring realistic aircraft separation constraints which might violate the triangle inequality.

To fill the gap of the existing research, a single algorithm is developed to find aircraft runway schedules that are optimal for both delay and throughput, to provide relatively fast computation times, and allows for realistic separation constraints.

This paper is organized as follows. In Section II a description of the runway scheduling problem is given along with a brief literature review. In Section III the basic dynamic programming algorithm is described. In Section III.E two heuristics are provided based off the dynamic programming approach, and in Section IV a set of results for the modeled Dallas/Fort Worth International Airport are presented. Finally, the authors close this paper in Section V with some lessons learned and possible extensions of the presented work.

II. Problem Statement and Background

In the following section, an overview of the problem is given, and relevant background information is provided. A comparison is made between techniques described in the literature and the proposed dynamic programming technique.

A. Problem Statement

Abstractly, the runway scheduling problem (RSP) can be thought of as a job shop scheduling problem [11] with precedence and release time constraints, where the objective is to sequence a set of jobs (aircraft) in a particular order to be processed by a processor (runway) so that some cost function is minimized. For the runway scheduling problem, one wants to find an efficient schedule for aircraft to use the runway. The output, then, to the runway scheduling problem is a time for each aircraft to begin using the runway, which will be referred to as a runway schedule.

The runway scheduling problem is formally described as follows: Given an ordered set of departure aircraft Q_d for each departure queue d, a set of departure aircraft G_g for each gate g of cardinality 1, an ordered set of aircraft C_c for each runway crossing queue c, a set of aircraft P_i that must not use the runway before aircraft i, an earliest time $\alpha(i)$ for aircraft i to reach the runway, and various timing constraints (described below), find the set of *non-dominating*[a] runway schedule(s) with respect to both delay and throughput.

A diagram of a typical problem is shown below in Figure 1. This diagram shows an example airport with one departure runway, a ramp area, and a network of taxiways. Each aircraft on the airport surface waits to use the runway in chain like structures called queues. For this diagram, departure aircraft located within the ramp area are at their gates or gate queues, and departure aircraft near the top left entrance of the departure runway are waiting in departure queues. Finally, there is one taxiway that crosses the runway with a chain of two aircraft waiting cross. These aircraft waiting to cross the runway are said to be waiting in their crossing queue.

To be exact, an aircraft a_j^i is indexed by its queue j and with its position denoted by i. The position count starts at 0 from the back of the queue and successively increases for aircraft closer to the front of the queue. In particular, this figure has two departure queues near the top of the runway, with 2 departure aircraft, a_0^1 and a_1^1, in departure queue 1, and 1 departure aircraft, a_0^2, in the departure queue 2. The figure also has 3 departure aircraft, a_0^3, a_0^4, and a_0^5, waiting at their gates (in the ramp area), and 2 aircraft, a_0^6 and a_1^6, waiting to cross the runway. Because aircraft in the same queue cannot simply pull in front of an aircraft waiting ahead of it, there will be precedence constraints between departure aircraft waiting in the same queues. To the same degree, departure aircraft located at the gates must wait to depart until their path is clear. Also since each aircraft i has only one $\alpha(i)$, it is assumed a single path to the runway is provided.

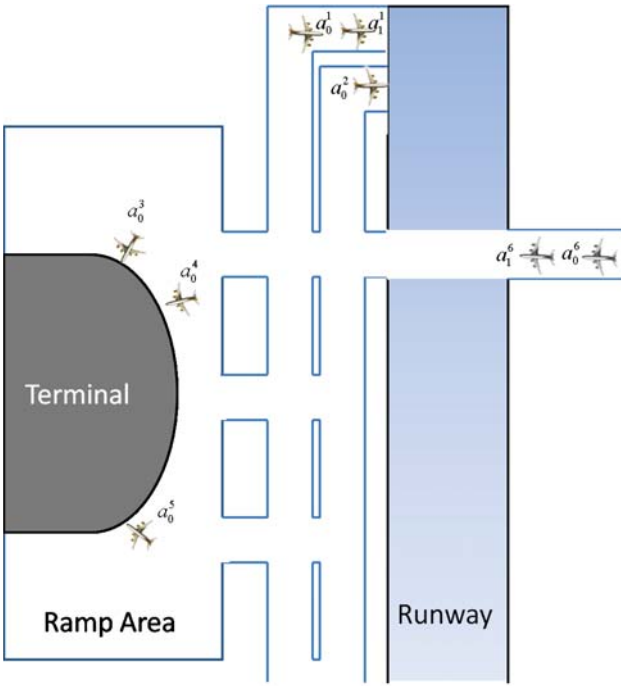


Figure 1. Problem Description

Based upon the various aircraft moving on the airport surface, a tower controller must figure out how to sequence the aircraft to best utilize the runway. Complicated timing constraints, however, make the task difficult.

[a] A set of non-dominating sequences with respect to throughput and delay are those that are either better at throughput or delay but not both. A formal understanding of this concept is given in section III.D.

1. **Departure Wake Vortex Separation:** Any departure aircraft following another aircraft to the same heading must wait a sufficient amount of time before departing due to the leading aircraft's wake vortex stream.
2. **Departure Area Navigation (RNAV) Separation:** If the departure aircraft has certain equipage requirements [12], a reduced wake vortex timing separation can be used as long as the trailing aircraft is going to a different heading.
3. **Runway Crossings Separation:** A departure wanting to take off must wait a minimum time to allow runway crossing traffic to clear. Vice versa, an aircraft crossing the runway must wait until a departure has cleared the runway before it can cross. Finally, for any two consecutive aircraft crossing the runway there is a minimum separation due to communication delay (e.g., voice clearances can not be given simultaneously to parallel crossings).
4. **Miles-in-Trail (MIT) Separation :** MIT restrictions are handed down from various Air Route Traffic Control Centers (ARTCC) and cause delays to ground departures to balance demand and capacity across the national airspace [13]. In this paper, spatial separations imposed by MIT restrictions are at a departure fix. These restrictions increase the time between two consecutive aircraft departing to the same fix. Moreover, the spatial separations are converted into a time equivalent based upon nominal aircraft speed.

While the above constraints are self explanatory for any two consecutive runway operations, one must be careful when considering more than two aircraft. For example consider Figure 2 below, where the inter-departure separation between consecutive departures is 60 seconds. If one only considers the consecutive separation values, then the second aircraft waits to depart 60 seconds after the time the first aircraft departs, and the third aircraft waits to depart 60 seconds after the time the second aircraft departs. However, given that the required separation between the first and the third aircraft is 218 seconds, by considering only the 60 second consecutive separation values one would not satisfy required aircraft separation between the first and the third aircraft. Since this situation can occur in reality, the authors do not ignore this possibility. For the remainder of the paper, the separations or constraints are said to violate the triangle inequality [14].

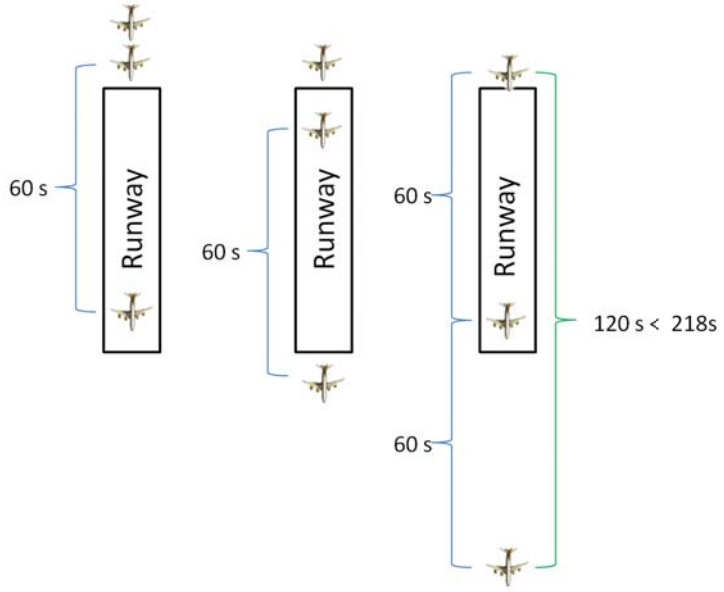


Figure 2. Triangle inequality violation.

A novel solution to this problem is given in Section III, but a brief review of the relevant literature is provided next.

B. Background

While there are several attempts to solve job shop scheduling problems reported in the literature, the authors focus on those attempts specifically developed to solve a runway scheduling problem. Often, a runway scheduling problem is solved by using either mixed integer linear programming or dynamic programming.

The runway scheduling problem can be formulated as a mixed integer linear program [4][5][6]. The authors in [4][6] show how all separation constraints can be incorporated and how departure queue assignments are possible. While mixed integer linear programs have the advantage of easily modeling a large class of optimization problems, their computation times are often unsuitable for real-time applications [8][10]. Alternatively, researchers have devised

methods for enabling faster computation of runway schedules through the use of dynamic programming techniques under Constrained Position Shifting (CPS) constraints.

In [7] Psaraftis shows how dynamic programming can be applied to solve a simplified runway scheduling problem by introducing additional constraints. Psaraftis uses CPS to limit deviations in the final position of an aircraft by comparing the final position to a reference runway schedule. Specifically, this is accomplished by constraining an aircraft to move a maximum number of positions from its reference position. For various reasons (equity, etc.), first-come-first-served is often taken as the reference sequence. Within the context of CPS, Psaraftis finds an optimal runway schedule for landing aircraft but does not consider any timing constraints which might violate the triangle inequality; therefore, this approach has limited application.

The dynamic programming approach in [7] provides a useful state definition which was then used by Rathinam et al. [10] in a generalized dynamic programming fashion to find a set of pareto optimal solutions with respect to both throughput and delay for departure aircraft only. Rathinam et al. extended the work done by Psaraftis for a runway scheduling problem by showing that it is necessary to keep track of the throughput while attempting to find a delay optimal aircraft runway schedules. While the authors in [10] present a fast solution for finding a set of pareto optimal aircraft runway schedules, they do not extend the formulation to incorporate necessary runway constraints where the triangle inequality is violated. For example, the authors in [10] only consider a problem where three departure queues are present and there exist no runway crossing operations, which are known to violate the triangle inequality. In contrast, this paper shows how any timing constraint can be incorporated using a similar state definition.

In [8] the authors show how using a dynamic programming method can solve the runway scheduling problem by developing a CPS network and then find an optimal makespan by using the dynamic programming method. The authors successfully incorporate all possible separation constraints and precedence constraints, but do not find a set of pareto optimal aircraft runway schedules, nor do they find the true optimal. Their solution, for example, includes CPS constraints and therefore finds sub-optimal either delay or throughput aircraft runway sequences, but not both. In contrast, this paper shows how a set of pareto optimal solutions can be calculated for with respect to delay and throughput without any consideration for CPS.

III. Generalized Dynamic Programming Approach

In this section a generalized dynamic programming method is given to solve a runway scheduling problem is described. In addition to finding optimal solutions, the authors also describe two efficient heuristics which use the basic dynamic programming framework.

A. State Definition

Psrafties provided the foundation for the state definition used for a simpler version of this problem, which was then also used by Rathinam et al. in [10]. In order to capture divergent heading operations not included by previous research efforts, an extension of the state definition is developed. A state for a runway scheduling problem is the number of aircraft yet to use the runway and the departure that just left to a particular departure heading.

Before showing the complete state definition, some notation is introduced. Any element l_i such that $i = \{1, ..., k_1\}$ are the number of remaining aircraft to depart from departure queue i. Any element l_i such that $i = \{k_1+1, ..., k_1+k_2\}$ are the number of remaining aircraft to depart from gate i. To continue, any element l_i such that $i = \{k_1 + k_2 + 1, ..., k_1 + k_2 + k_3\}$ are the number of remaining aircraft to cross the runway from runway crossing queue i. Finally, m_i belongs to the set $\{1, ..., k_1 + k_2 + k_3\}$ indicating the queue that the last departure aircraft which left to heading $i = \{1, ..., n\}$. Formally then, a state S is defined as,

$$S = (l_1, l_2, .., l_{k_1}, l_{k_1+1}, ..., l_{k_1+k_2}, l_{k_1+k_2+1}, ..., l_{k_1+k_2+k_3}, m_1, m_2, ..., m_n) \tag{1}$$

For the remainder of this paper, the authors assume that the initial state S_o has q_i aircraft remaining to use the runway for $i = \{1, ..., k_1 + k_2 + k_3\}$ and that if no aircraft have departed to heading k at state S then $m_k = -1$. To reduce unnecessary notation, let $K = k_1 + k_2 + k_3$, and without loss of generality the authors assume that there are only two headings based upon operational practice. Furthermore, a state S with a corresponding sequence history s (a sequence of aircraft which have used the runway) is denoted by S_s.

B. Precedence Constraints and Sample Recursion

To completely enumerate the feasible solution space (e.g., the possible runway schedule(s)), all precedence relationships must be handled appropriately while recursively branching from state to state. In practice precedence constraints can be priority based, but the only precedence considered here are those which arise due to physical conflicts on the airport surface. These constraints can easily be determined by observing an aircraft's route and the queuing structures.

To show how precedence constraints are built and respected through the state transformation, consider Figure 3. This figure shows a mock-up airport with various aircraft waiting to use the runway. This airport assumes two

departure headings, and for sake of simplicity each departing aircraft is supposed to go to heading one. The state is given next to each picture, indicating how the state transformations occur. The first stage of the recursion from the initial state shown in Figure 3 is given in Figures 4(a), 4(b), and 4(c). Notice that the departure aircraft at the gate (shown in the ramp area) can never use the runway because there is always at least 1 departure aircraft blocking its path to the runway. In contrast, aircraft that are going to cross the runway do not need to wait until the aircraft in the departure queue are cleared.

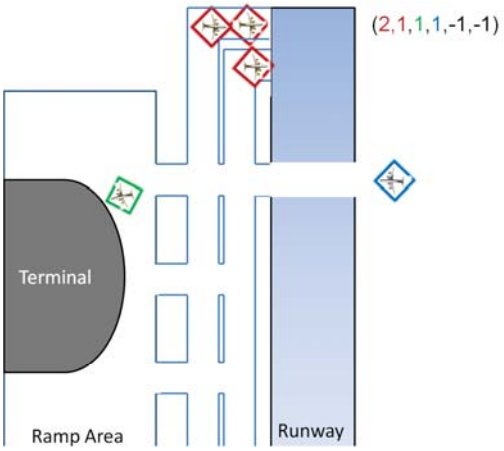
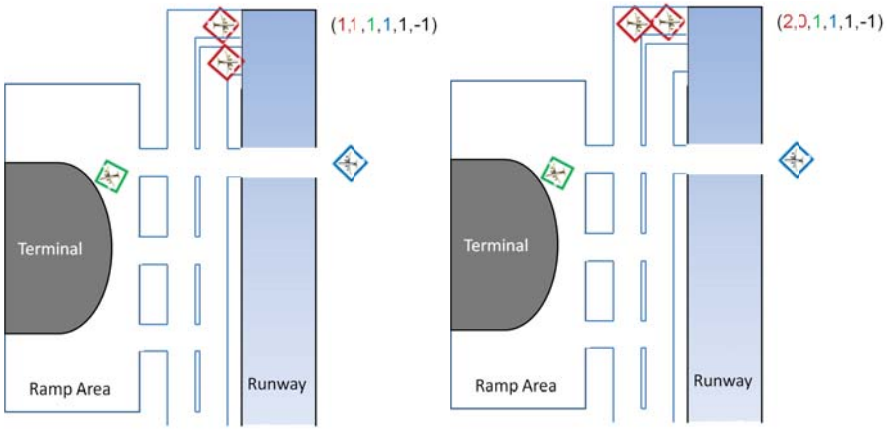
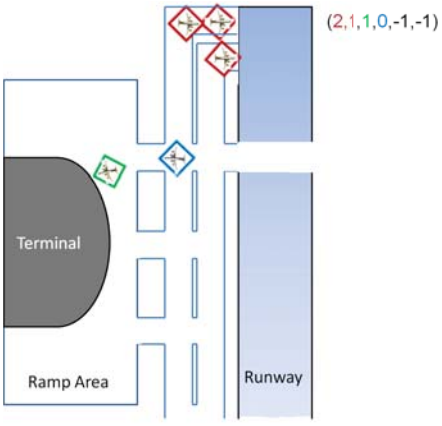


Figure 3. Initial state and Parent



(a) Child 1 (b) Child 2



(c) Child 3

Figure 4. First stage in recursion.

To completely enumerate the state space, one simply needs to continue branching from each child state. When two sequences converge to the same state, their cost functions need to be checked in order to verify the dominance of one sequence over the other. These cost functions are described next, and their comparison is given in Section III.D.

C. Timing Constraints using Supporting Cost Functions

For each time based constraint, a supporting cost function is required. While the authors only consider wake vortex separation, miles-in-trail requirements, and runway crossings, the ideas presented in this section suggest a general methodology to account for any time based separation between aircraft. In this section, the authors provide a mathematical description for each supporting cost function.

1. Wake Vortex Separation

To successfully include all variations of wake vortex separation, two separation matrices are provided in Tables 1 and 2. These matrices are a product of an empirical study of surveillance data. In particular, Table 1 represents the separation required between two consecutive departure aircraft going to the same heading and Table 2 represents the separation required between two consecutive departure aircraft going to different headings. Table 2 can only be used when the consecutive aircraft are RNAV equipped. Usually, however, most aircraft are RNAV equipped. For example, approximately 94% of aircraft at DFW are RNAV equipped [12].

Table 1. Inter-departure aircraft separation in seconds. Column aircraft lead row aircraft.

	Small	Large	Heavy-B757
Small	45	67	80
Large	45	67	80
Heavy-B757	45	67	67

Table 2. Inter-departure RNAV separation in seconds. Column aircraft lead row aircraft.

	Small	Large	Heavy-B757
Small	40	59	80
Large	40	41	80
Heavy-B757	40	41	67

To ensure that separation is satisfied using the above wake vortex tables, one must keep track of the last time $H_h^{last}(S_s)$ a departure departed for each heading h at a given S_s. This value can be calculated for each heading $h = \{1, 2\}$ by the following recursion,

$$H_h^{last}((l_1, ..., l_K, m_1, m_2)_s) = \begin{cases} -N & \text{if } m_h = -1 \\ H_h^{last}((l'_1, ..., l'_K, m'_1, m'_2)_{s'}) & \text{if } l'_{m_h} = l_{m_h} \\ \text{NULL} & \text{if } l'_{m_h} \neq l_{m_h} \text{ and } \wp \\ TIME((l_1, ..., l_K, m_1, m_2)_s) & \text{otherwise} \end{cases} \tag{2}$$

where for each $k = \{1, ..., K\}$

$$l_k = \begin{cases} l'_k - 1 & \text{if an aircraft just left queue } k \\ l'_k & \text{otherwise} \end{cases} \tag{3}$$

and for each $h = \{1, 2\}$

$$m_h = \begin{cases} k & \text{if a } \textit{departure} \text{ aircraft just left queue } k \text{ and went to heading } h \\ m'_h & \text{otherwise} \end{cases} \tag{4}$$

and $\wp$ represents the following condition

$$\wp = \begin{cases} true & \text{if } \exists j \in P_{a_{l_{m_h}}^{m_h}} \text{ such that } U(j) = false \\ false & \text{otherwise} \end{cases} \tag{5}$$

where $U(j)$ is a binary function that is equal to $false$ if and only if aircraft j has not used the runway yet.

$TIME((l_1, ..., l_K, m_1, m_2)_s)$ indicates the time an aircraft last used the runway at state $S = (l_1, ..., l_K, m_1, m_2)$ from sequence history s. A recursion for $TIME((l_1, .., l_K, m_1, m_2)_s)$ is given later, but simply knowing its meaning is sufficient for the current discussion.

From the above equations, one can see that the last time $H_h^{last}(S_s)$ an aircraft used heading h is either equal to a large negative number $-N$ if no aircraft has departed to heading h, equal to the time that the last departure departed to heading h stored from the prior iteration ($S'_{s'}$) if the last aircraft to use the runway did not go to heading h, $NULL$ if a precedence constraint is violated, or time $TIME(S_s)$ if the last aircraft to use the runway did go to heading h.

2. Miles-in-trail

Miles-in-trail (MIT) restrictions can often arise due to convective weather from various locations throughout the National Airspace System. In airspace local to airports, these restrictions may be realized by limiting the departure rate to a particular departure fix by requiring a relatively large separation between consecutive departure aircraft to that fix. For example, at Dallas/Fort Worth International Airport there are 16 departure fixes around the airport's local airspace, where departure aircraft will be routed through depending on their destination airport. For the interested reader, MIT restrictions are covered in some detail in references [15] and [13].

To incorporate an MIT constraint for departure aircraft heading to the same departure fix f, the last time $F_f^{last}(S_s)$ a departure aircraft used fix f must be tracked for each S_s. First assume that $fix(a)$ is aircraft a's corresponding fix, then $F_f^{last}(S_s)$ can be updated for each fix f with the following recursion,

$$F_f^{last}((l_1, ..., l_K, m_1, m_2)_s) = \begin{cases} -N & \text{if } m_1 = m_2 = -1 \\ \text{NULL} & \text{if } l_{m_i} \neq l'_{m_i} \text{ and } \wp \\ TIME((l_1, ..., l_K, m_1, m_2)_s) & \text{if } \exists i\text{: } fix(a_{l_{m_i}}^{m_i}) = f,\ l_{m_i} \neq l'_{m_i}, \text{ and } \neg\wp \\ F_f^{last}((l'_1, ..., l'_K, m'_1, m'_2)_{s'}) & \text{otherwise} \end{cases} \tag{6}$$

where l'_k, m'_h, $\wp$, and $TIME((l_1, ..., l_K, m_1, m_2)_s)$ have the same meaning as before.

The above recursion updates the time $F_f^{last}(S_s)$ an aircraft last left to departure fix f to $TIME(S_s)$ if a departure just used fix f. If the aircraft to just use the runway did not go to fix f however, then $F_f^{last}(S_s)$ is either equal to the prior state's value $F_f^{last}(S'_{s'})$, or a large negative number $-N$ (e.g., no aircraft has left to this fix yet). Finally, if a there is a precedence violation, then $F_f^{last}(S_s)$ is simply equal to $NULL$ indicating an impossible state.

3. Runway Crossings

In addition to the above constraints, runway crossings are also accounted for. To successfully incorporate runway crossings two timing values need to be tracked, the time the last departure departed $D^{last}(S_s)$ and the time the last runway crossing occurred $C_c^{last}(S_s)$ from runway crossing c at S_s. It can easily be seen that,

$$D^{last}(S_s) = \max_{h=\{1,2\}} \{H_h^{last}(S_s)\} \tag{7}$$

Furthermore, we can calculate the time the last runway crossing occurred $C_c^{last}(S_s)$ for each runway crossing queue c with the following recursion,

$$C_c^{last}((l_1, ..., l_K, m_1, m_2)_s) = \begin{cases} -N & \text{if } l_c = q_c \\ TIME((l_1, ..., l_K, m_1, m_2)_s) & \text{if } l'_c \neq l_c \\ C_c^{last}((l'_1, ..., l'_K, m'_1, m'_2)_{s'}) & \text{otherwise} \end{cases} \tag{8}$$

where $TIME((l_1, ..., l_K, m_1, m_2)_s)$, l'_k, and m'_h have the same meaning as before.

D. Main Cost Functions and Non-Dominating Solutions

For the runway scheduling problem, delay and throughput are the primary or main cost functions. While the time values given in equations (2), (6), (7), and (8) are necessary to find optimal throughput and delay, they are simply supporting values to achieve the optimal main cost function values. This section first provides the mathematical

recursion to find the main cost function values, throughput ($TIME()$) and delay ($DELAY()$). This section then concludes by giving a condition for which all dual-optimal solutions must maintain.

1. Main Cost Functions

THROUGHPUT. As stated before, $TIME(S_s)$ represents the last time an aircraft used the runway at state S with corresponding sequence history s. This value is also the throughput or makespan of the runway and can be calculated for recursively. Before showing its recursion, various time constants must be introduced:

1. Sep_c indicates the time an aircraft takes to cross the runway at runway crossing c.
2. MIT_f indicates the time separation between consecutive departures to fix f.
3. $R[x][y]$ indicates the time separation between consecutive departures going to different headings with weight class x following weight class y
4. $Sep[x][y]$ indicates the time separation between consecutive departures going to the same heading with weight class x following weight class y
5. DEP is the estimated amount of time a departure takes to clear the runway.
6. X is the amount of time separation required between two consecutive runway crossing from the same crossing queue.

In addition to the above values, $M(a)$ is the weight class (Large, Heavy, Small) of aircraft a. Then, $TIME(S_s)$ can be calculated in the following manner (assuming all values are not $NULL$),

For each case (A-D) a description is given,

1. Case A: No aircraft has used the runway yet.
2. Case B: Departure aircraft at S_s just departed to heading 1 and fix f.
3. Case C: Departure aircraft at S_s just departed to heading 2 and fix f.
4. Case D: Aircraft at S_s just crossed the runway at crossing c.

$$TIME(S_s) = \begin{cases} 0 & \text{if Case A} \\ \max\{TIME(S'_{s'}) + \max\{\max_c\{C_c^{last}(S'_{s'}) + Sep_c\}, \\ F_f^{last}(S'_{s'}) + MIT_f, H_1^{last}(S'_{s'}) + Sep[M(a_{l_{m_1}}^{m_1})][M(a_{l'_{m'_1}}^{m'_1})], \\ H_2^{last}(S'_{s'}) + R[M(a_{l_{m_1}}^{m_1})][M(a_{l'_{m'_2}}^{m'_2})]\}, \alpha(a_{l_{m_1}}^{m_1})\} & \text{if Case B} \\ \max\{TIME(S'_{s'}) + \max\{\max_c\{C_c^{last}(S'_{s'}) + Sep_c\}, \\ F_f^{last}(S'_{s'}) + MIT_f, H_1^{last}(S'_{s'}) + R[M(a_{l_{m_2}}^{m_2})][M(a_{l'_{m'_1}}^{m'_1})], \\ H_2^{last}(S'_{s'}) + Sep[M(a_{l_{m_2}}^{m_2})][M(a_{l'_{m'_2}}^{m'_2})]\}, \alpha(a_{l_{m_1}}^{m_1})\} & \text{if Case C} \\ \max\{TIME(S'_{s'}) + \max\{D^{last}(S'_{s'}) + DEP, C_c^{last}(S'_{s'}) + X\}, \\ \alpha(a_{l_c}^{c})\} & \text{if Case D} \end{cases} \tag{9}$$

DELAY. To successfully add delay one simply needs to track the delay at each state S_s for each corresponding sequence s. Delay is calculated as,

$$DELAY(S_s) = \begin{cases} 0 & \text{if } l_i = q_i \text{ for all } i \in \{1, .., K\} \\ DELAY(S'_{s'}) + TIME(S_s) - \alpha(a_{l_k}^k) & \text{if last aircraft left from queue } k \end{cases} \tag{10}$$

2. Non-dominating Solutions

Next, the authors provide a set of conditions that guarantee the optimality of a runway schedule. To start, two sequences (e.g., s and s') can only be compared against each other once they arrive at an identical state S. Then, to guarantee that a sequence s is non-dominated by any other sequence s' one must be sure that the following condition is true,

For each $s' \neq s$

$$\begin{array}{lll} \text{for each } c = \{k_1 + k_2 + k_3 + 1, ..., K\} & C_c^{last}(S_s) < C_c^{last}(S_{s'}) & \text{or} \\ \text{for each } f = \{1, ..., FIX\} & F_f^{last}(S_s) < F_f^{last}(S_{s'}) & \text{or} \\ \text{for each } h = \{1, 2\} & H_h^{last}(S_s) < H_h^{last}(S_{s'}) & \text{or} \\ & TIME(S_s) < TIME(S_{s'}) & \text{or} \\ & DELAY(S_s) < DELAY(S_{s'}) & \end{array} \tag{11}$$

If, for example, sequence s maintains values less than or equal to s' for all time values being compared above, then we say that s dominates s'. In this way, one would remove sequence s' at state S from the possible solutions because it is not-optimal. An inductive proof can be constructed showing that the aforementioned generalized dynamic programming technique satisfies conditions that guarantee optimality. [16][17].

E. Heuristics

By taking advantage of CPS constraints to reduce the search space, two efficient heuristics to solve a runway scheduling problem are presented. Both heuristics are formulated using the following 2-step process:

1. The first step tries to find a good reference sequence s_{ref}. To accomplish this, the initial state S_o is reconfigured by artificially creating precedence constraints between aircraft at the gates.
2. The second step is a refinement process where the reference sequence s_{ref} is improved with respect to throughput and delay. To accomplish this, s_{ref} is refined by re-solving the original problem (e.g., S_o) and by inducing CPS constraints for each *departure* aircraft's position with respect to s_{ref}.

To show this two step process, the authors explain how the first heuristic is formulated. The first heuristic reduces the search space (e.g., number of possible sequences) by sorting departure aircraft at gates in first-come-first-served virtual queues based upon aircraft weight class (step 1). Aircraft at gates of the same weight class are combined into a single queue on a first-come-first-served basis, regardless of their other type-parameters (e.g., fix and heading). After recombining gate aircraft into virtual queues, one solves the new problem with the redefined state using the algorithm described above. After solving this problem, a solution is chosen to become s_{ref} for use in step 2.

In step 2, the reference sequence s_{ref} is improved by using CPS constraints. To achieve this, the initial state S_o is solved using the generalized dynamic programming approach with CPS constraints on *departure* aircraft and s_{ref} is used to define the CPS reference positions.

The second heuristic is similar to the first, but instead the reference sequence (step 1) is found by forming virtual queues based upon departure heading instead of weight class.

For the remainder of this paper, including all figures and charts, the heuristic where weight is the deciding factor for redefining the initial state will be referred to as the *Weight Class Heuristic* (WCH). For similar reasons, the second heuristic will be referred to as the *Heading Heuristic* (HH).

IV. Simulation Results

In this section, simulation setup and results are given. The results show the computational time and quality of runway schedules obtained by finding optimal and heuristic solutions.

A. Simulation Setup

For the following results a model of the east side of Dallas/Fort Worth International Airport in a south flow configuration is used. Runway 17R is used as the departure runway which contains 3 departure queues, 2 departure headings, 12 departure fixes, and 5 runway crossing queues. In addition, over 20 gates are used for the simulation.

All parameters were gathered empirically using ASDE-X [18][**?**] surface surveillance data. The standard wake vortex separation and reduced wake vortices separation for large, heavy, and small aircraft are given in Tables 1 and 2, respectively. For all simulated trials, only 1 fix had a miles-in-trail separation of 218 seconds. The authors assume that it takes a departure 30 seconds to clear the runway, and it takes 14 seconds for a plane to cross the runway. Finally, an aircraft crossing the runway will have 6 seconds separation between them (e.g., $X = 6$), irrespective of crossing queue. This feature is to account for a time-lag when controllers issue voice clearances[b].

The authors consider 3 different baselines for this study. It is well known that tower controller strategies vary, and so it is only fair to assume that controllers will respond to their tasks differently based on their experience and skills. These baselines are summarized below,

1. FCFS-CPS-1 : This heuristic uses a first-come-first-served reference sequence with a CPS for departure aircraft of at most 1
2. FCFS-CPS-3 : This heuristic uses a first-come-first-served reference sequence with a CPS of position for departure aircraft of at most 3.
3. FCFS-CPS-5 : This heuristic uses a first-come-first-served reference sequence with a CPS for departure aircraft of at most 5.

Aircraft are uniformly distributed to fixes, headings, and weight class. The total number of aircraft is split between the number of aircraft crossing the runway and the number of aircraft departing off the runway. Traffic loads ranged from 10 to 30 in increments of 2, and therefore, there are eleven different traffic loads. All aircraft are able

[b]Effectively, this reduces the number crossing queues to 1. See [7].

to reach the runway within 15 minutes so that each simulation is relatively dense. For each of the 11 traffic loads, there are 100 uniformly distributed instances. In total then, there are 1100 (11×100) runs for each algorithm (2 heuristics[c], 1 optimal, 3 baselines)[d].

B. Results

Since there could be more than one solution in the set of non-dominated solutions, a thorough study should analyze all solutions and their various characteristics. For the purposes of this study however, one solution was chosen. For the optimal and heuristics, the solution chosen was the one that contributed to best delay. In contrast, controllers are often concerned with best throughput, and therefore, the best throughput solution was chosen for all baselines.

First, an analysis of delay savings is given. The main purpose is to understand how much delay, measured as the excess time over $\alpha(i)$, was attributed by each algorithm. To compare how well each of the heuristics and optimal performed over the baselines, the expected delay per aircraft is given in Table 3. This table shows that the optimal achieves a 40 to 70 percent decrease in the expected delay per aircraft depending on the baseline its compared against. In addition, the heuristics show similar results compared to the baselines and almost have the same expected delay per aircraft as the optimal.

Table 3. Delay Per Aircraft

	Optimal	WCH	HH	FCFS-CPS-1	FCFS-CPS-3	FCFS-CPS-5
Avg. Delay (sec) per Aircraft	118.70	124.53	120.69	365.84	266.07	209.02
%Decrease over FCFS-CPS-1	67.56	67.01	65.96			
%Decrease over FCFS-CPS-3	55.34	54.64	53.20			
%Decrease over FCFS-CPS-5	43.21	42.26	40.42			

Next, an analysis is given to illustrate the benefit of the heuristics with respect to increased traffic loads. Figure 5(a), for example, shows the difference in delay attributed by the Weight Class Heuristic less that of the optimal. Furthermore, Figure 5(b) shows the difference in the delay attributed by the Heading Heuristic less that of the optimal. Since there are 100 runs (independent axis) for each traffic load, the results show how well the heuristics perform as the traffic load progressively increases. For example, runs 1-101 are for a traffic load of 10 aircraft, 102-201 are for a traffic load of 12, etc... Therefore, when the number of aircraft for a given run is low, the heuristics perform well; however, when the number of aircraft is high, these heuristics perform increasingly worse.

To continue, the Weight Class Heuristic shows a maximum deviation in optimal delay of 1700 seconds approximately, and the Heading Heuristic shows a maximum deviation in optimal delay of about 590 seconds. While it appears the Heading Heuristic out performs the Weight Class Heuristic, this is likely a function of the inputs. A more precise study could help to understand the scenarios where one heuristic outperforms the other.

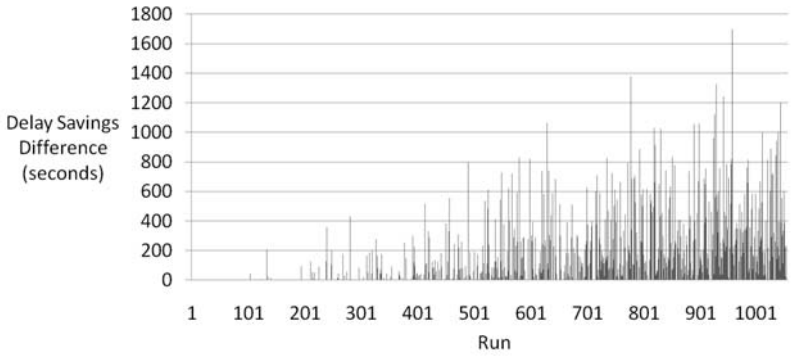
In order to determine how well the optimal, heuristic(s), and baseline(s) perform with respect to throughput, the authors show the expected number of runway operations per hour in Table 4. The optimal and heuristics achieve an average of approximately 63 runway operations per hour, whereas the best baseline (FCFS-CPS-5) achieves an average of approximately 62 runway operations per hour. For the optimal and heuristics, there is approximately an %8 increase of throughput over FCFS-CPS-1 and approximately a %1 increase over FCFS-CPS-5. This suggests that the heuristic(s) and optimal solution perform better than all three baselines with respect to throughput on average.

Table 4. Number of Runway Operations

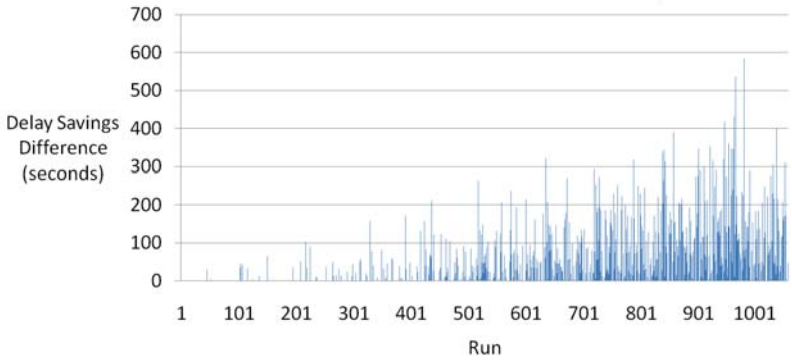
	Optimal	WCH	HH	FCFS-CPS-1	FCFS-CPS-1	FCFS-CPS-1
Avg. # of Runway Ops. per Hour	63.01	62.81	62.92	58.11	61.21	62.23
%Inc. over FCFS-CPS-1	8.40	8.12	8.13			
%Inc. over FCFS-CPS-3	2.99	2.61	2.63			
%Inc. over FCFS-CPS-5	1.31	1.07	1.08			

[c]Note, during the refinement process (step 2) for the heuristics a CPS of 5 was used.

[d]Unfortunately, when there are 30 aircraft present, the computer system is not capable of storing the required memory to solve for the optimal solution, resulting in 1033 runs (67 runs were not completed). Memory is relatively cheap, and therefore, acquiring additional memory proves to be no challenge if solving larger problem instances is desired.



(a) Weight Class Heuristic



(b) Heading Heuristic

Figure 5. Difference in delay between optimal solution and heuristic solution

Finally, a comparison of the computational effort of the heuristics and optimal is given below. Figures 6, 7, and 8 indicate the computation time (in seconds) to complete each run for the optimal, Weight Class Heuristic, and Heading Heuristic, respectively. It's not surprising that the computation time increases rapidly to find the optimal solution as the number of aircraft increases. The computation time for the Weight Class Heuristic increases marginally up to 2.5 seconds from close to 0 seconds for smaller instances (10 aircraft). Moreover, the computation time for the Heading Heuristic increases from approximately 0 seconds for smaller instances up to .17 seconds for larger instances. These results suggest that both heuristics would be suitable for real time application.

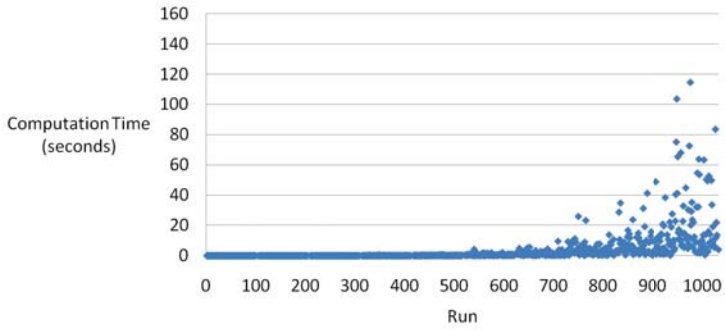


Figure 6. Computation times using the optimal.

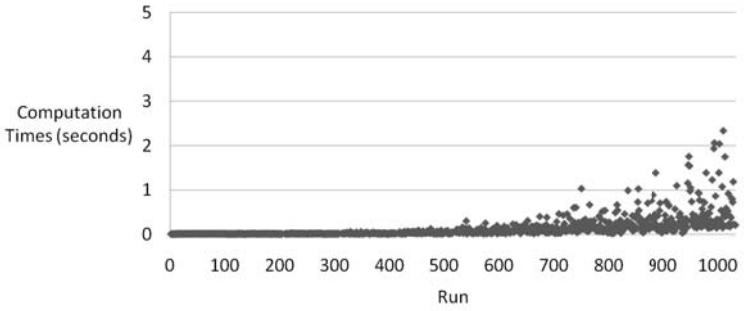


Figure 7. Computation times using the Weight Class Heuristic.

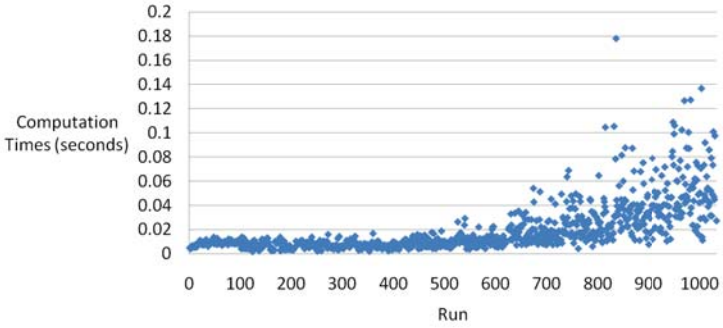


Figure 8. Computation times using the Heading Heuristic.

V. Conclusion

A generalized dynamic programming method was presented to solve the runway scheduling problem. The algorithm constructed from the generalized dynamic programming technique differs from other formulations because it can find the set of non-dominated optimal solutions with respect to delay and throughput while allowing constraints that violate the triangle inequality . Two new heuristics based on the generalized dynamic programming technique are presented that show suitable computational performance and solution quality for real-time applications. The heuristics and optimal outperform 3 baselines which attempt to model different controller strategies. The heuristics come within 5% of the optimal's delay per aircraft and achieve approximately the same throughput.

For future studies, it will be important to expand the current tool to incorporate aircraft on the taxiway. To accomplish this, the authors believe a taxi scheduler will need to be integrated as part of the final solving technique. When taxi conflicts arise on the airport surface they can hurt the runway schedulers solution or completely cause the solution to be infeasible. In particular, since the runway scheduler does not account for taxiway conflicts, certain sequences it suggests could be infeasible or completely unreasonable to practice. In addition to adding a taxi scheduling element, uncertainty analysis needs to be conducted to determine algorithmic robustness. For example, it seems unlikely that the runway schedules found from the presented algorithms will be followed precisely due to uncertainties in $\alpha(i)$, and therefore doing an uncertainty analysis will help to provide a meaningful interpretation of the benefits presented in this paper. Since the application of this algorithm is at a tactical level of scheduling, rolling planning horizon algorithms need to be implemented to mend solutions from consecutive schedule calls. Finally because these algorithms could easily be expanded to additional airport systems, one could provide a deeper analysis of algorithmic performance across many airport systems.

References

[1] S. Zelinkski and T. Romer, "An Airspace Concept Evaluation System Characterization of National Airspace System Delay," AIAA 4th Aviation Technology, Integration and Operations (ATIO), September 2004.

[2] Idris, H., Delcaire, B., Anagnostakis, I., Hall, W., Pujet, N., Feron, E., Hansman, R., Clarke, J., and Odoni, A., "Identification of Flow Constraint and Control Points in Departure Operations at Airport Systems," AIAA Guidance, Navigation, and Control Conference, Boston, MA, Aug. 10-12, 1998.

[3] Wood Z., Kistler M., Rathinam S., and Jung Y., "A Simulator for Modelling Aircraft Surface Operations at Airports," AIAA Modeling and Simulation Technologies Conference, Chicago, IL., August 2009.

[4] Gupta, G., Malik, W., and Jung, Y. "Incorporating Active Runway Crossings in Airport Departure Scheduling," AIAA Guidance, Navigation and Control Conference, Toronto, Canada, August 2-5, 2010.

[5] Brinton, C., Atkins, S., Cook, L., Lent, S., and Prevost, T., "Ration by Schedule for airport arrival and departure planning and scheduling," Integrated Communications Navigation and Surveillance Conference (ICNS), Herndon, VA, 11-13 May 2010.

[6] Gupta, G., Malik., W., and Jung, Y., "A Mixed Integer Linear Program for the Airport Departure Scheduling Problem," 9th AIAA Aiviation Technoology, Integration, and Operations Conference (ATIO), Hilton Head, SC, September 2009.

[7] H.N., Psaraftis, "A dynamic programming approach for sequencing groups of identical jobs," Operations Research 28 (1980), 13471359.

[8] Balakrishnan, H. and Chandran, B., "Efficient and Equitable Departure Scheduling In Real-Time: New Approaches To Old Problems," 7th USA/Europe Air Traffic Management R&D Seminar, July 2007, Barcelona, Spain.

[9] Stiverson, P., Rathinam, S., "Heuristics for a runway-queue management problem," Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, August 2010.

[10] 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, Illinois, August 10-13, 2009.

[11] M.R. Garey, D.S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," Mathematics of Operations Research, 1976, pp. 117-129.

[12] Borchers, P. and Day, K., "Analysis OF Divergences From Area Navigation Departure Routes at DFW Airport," 28^{th} Digital Avionics Systems Conference, Orlando, FL, October 2009.

[13] Grabbe, S., and Sridhar., B., "Modeling and evaluation of Miles-in-Trail restrictions in the National Air Space," AIAA Guidance, Navigation and Control Conference, Austin, TX, August 2003.

[14] Ruding, W., "Principles of Mathematical Analysis," McGraw-Hill, ISBN 007054235X, 1976.

[15] Rios, J., "Aggregate Statistics of National Traffic Management Initiatives," AIAA 10th Aviation Techonology, Innovation and Operations (ATIO) Forum, Fort Worth TX, September 2010.

[16] Bellman, R.E., "Dynamic Programming," Princeton University Press, Princeton, NJ, Republished 2003, ISBN 0486428095.

[17] Carraway, R.L., and Morin, T.L., "Theory and applications of generalized dynamic programming: an overview," International Journal of Computers and Mathematics with Applications, 16, 779-788, 1988.

[18] Airport Surface Detection Equipment Model X, Sensis Corporation. [http://www.sensis.com/docs/128]