16th AIAA Aviation Technology, Integration, and Operations Conference

Exact and Heuristic Algorithms for Runway Scheduling

Waqar A. Malik*
University of California Santa Cruz, NASA Ames Research Center, Moffett Field, CA, 94035, USA

Yoon C. Jung†
NASA Ames Research Center, Moffett Field, CA, 94035, USA

This paper explores the Single Runway Scheduling (SRS) problem with arrivals, departures, and crossing aircraft on the airport surface. Constraints for wake vortex separations, departure area navigation separations and departure time window restrictions are explicitly considered. The main objective of this research is to develop exact and heuristic based algorithms that can be used in real-time decision support tools for Air Traffic Control Tower (ATCT) controllers. The paper provides a multi-objective dynamic programming (DP) based algorithm that finds the exact solution to the SRS problem, but may prove unusable for application in real-time environment due to large computation times for moderate sized problems. We next propose a second algorithm that uses heuristics to restrict the search space for the DP based algorithm. A third algorithm based on a combination of insertion and local search (ILS) heuristics is then presented. Simulation conducted for the east side of Dallas/Fort Worth International Airport allows comparison of the three proposed algorithms and indicates that the ILS algorithm performs favorably in its ability to find efficient solutions and its computation times.

I. Introduction

Meeting the projected increase in air traffic demand within the National Airspace System (NAS) requires improvements in all areas of air traffic management. Airports, being the origin or destination of the air traffic network, encounter some of the highest traffic density in the NAS. During peak periods at major airports, capacity limitations on the airport surface area create bottlenecks and cause delays to both departures and arrivals. This congestion effect and the associated delays persist for a significant part of the peak period, and often restrict an airport's throughput by hampering runway operations. Idris et al.[1] observed that a majority of airport surface delay was incurred at the runways.

The observed congestion and delay at the airport runways have spurred recent research in finding efficient solutions for runway use. For the general runway scheduling problem, these solutions specify the schedule for each aircraft to use the runway: the wheels-off times for departing aircraft, the wheels-on times for arriving aircraft and the crossing times for aircraft that need to cross an active runway. Moreover, the solutions to the runway scheduling problem have to satisfy numerous physical and operational constraints, such as wake-vortex separation for successive departures, miles-in-trail restrictions over certain departure fixes, and time-window constraints for some departure aircraft. The runway scheduling problem also depends on the layout of the runway system: the operations on parallel runways (that are close together) or intersecting runways must be coordinated, and the actual separation requirements depend on the exact layout and use of the runways.

The runway scheduling problem is structurally equivalent to a job shop scheduling problem.[2-4] The runways represent the *machines*, and the aircraft represent the *jobs*. The required separation times between pairs of aircraft on the same runway are the (sequence dependent) *processing times*. The earliest possible time an aircraft can use the runway represents the release time of the job and the latest, the due date. A common objective is to minimize the completion time (runway-use time) of the last job, which is equivalent to maximizing throughput. Hence, many of the solution techniques commonly used for solving the job shop scheduling problems have been adapted to the runway scheduling problem; e.g., mixed integer linear programs,[5-6] branch and bound,[7] dynamic programming,[8-13]

* Research Scientist, University Affiliated Research Center, MS 210-8, Moffett Field, CA 94035.
† Aerospace Engineer, NASA Ames Research Center, MS 210-6, Moffett Field, CA 94035, AIAA senior member.

heuristics,[14-16] metaheuristics,[17-18] and others. The sequence dependent job shop scheduling problem is strongly NP-hard, and consequently, it is not expected to find polynomial time algorithms for the runway scheduling problem.

The majority of the prior papers on runway scheduling have looked at subsets of the general runway scheduling problems: the single runway scheduling for arrivals (arrivals scheduling problem) or departures (departures scheduling problem). Researchers have also made several simplifications to the problem to make it computationally tractable. Bianco et al.[4] have relaxed the wake turbulence separation criteria and scheduling between successive aircraft is based on a constant separation time rather than separation based on aircraft weight class. Researchers have also proposed the idea of constrained position shifting[14] that limits the number of positions an aircraft can occupy in a sequence. This reduces the available solution space and leads to computationally tractable solutions. Bennell et al.[19] provide a comprehensive review of airport runway scheduling.

Over the last few years, NASA Ames Research Center has developed the Spot and Runway Departure Advisor (SARDA) concept as a Decision Support Tool (DST) for surface management. The SARDA concept provides aircraft specific sequence and time advisories to Air Traffic Control Tower (ATCT) controllers to reduce the number of aircraft on the taxiways and runway queues while maintaining maximum throughput. This initial concept was demonstrated and evaluated in human-in-the-loop simulations for the east side of the Dallas/Fort Worth International Airport (DFW) with retired ATCT controller participants in 2010[20] and 2012.[21] The runway scheduling algorithm is the primary component of the SARDA concept since it determines the runway times that drive the computation of spot release or gate pushback times.

This paper looks at the Single Runway Scheduling (SRS) problem with arrivals, departures, and crossing aircraft on the airport surface. Constraints for wake vortex separations, departure area navigation (RNAV) separations and take-off time-window for departures are explicitly considered. We develop a multi-objective dynamic programming (DP) based algorithm that finds the optimal (exact) solution to the SRS problem. Given the exponential complexity of the exact DP algorithm, we then employ heuristics in the DP algorithm to find computationally efficient solutions to the SRS problem. We finally present an algorithm based on a combination of insertion and local search heuristics and compare it with the DP based algorithms.

II. Problem Setup

The modeling of the runway scheduling procedure depends on the taxiway layout, availability of holding area, number of spots (entry points to movement area) and runway configuration. In this paper, we consider the Single Runway Scheduling (SRS) problem. To better explain the SRS problem, consider the illustration provided in Figure 1, which depicts the east side of DFW as an example.

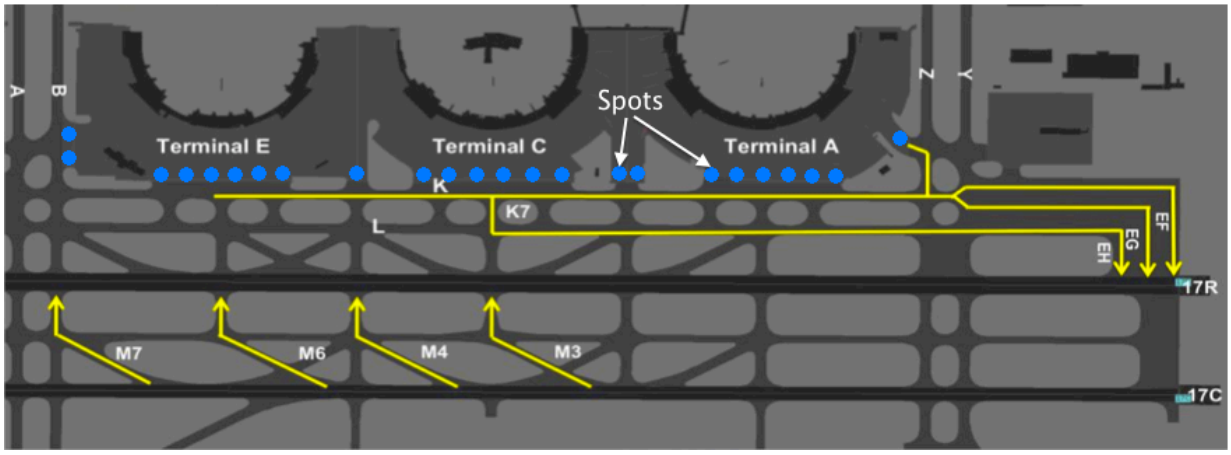

Figure 1. Airport Layout: runway 17R is the runway being scheduled. Arrivals land on 17C and hold at 17R for crossing. Departures approach runway 17R joining one of the three queues (EF, EG, and EH).

As depicted in the illustration, the SRS algorithm will be applied to control the use of runway 17R by the departure, crossing and arrival aircraft[‡]. Arrivals land on 17C, take one of the four exits (M3, M4, M6, M7), and arrive at the 17R crossing. The SRS algorithm provides the schedule for landing or crossing, though there is a limited time window to schedule the landing through vectoring and speed control in the terminal airspace. In this

[‡] At DFW, runway 17R is not used for arrivals.

paper, we assume that the relative sequences of arrivals cannot be changed. The ramp controller (airline or airport authority) clears the departure aircraft from the gate to the spot. The "spots" are physical regions on the airport (shown in blue circles) where control of the aircraft is transferred from the ramp controllers to the FAA Air Traffic Control Tower (ATCT) controllers. Once released from the spots, the departure aircraft travels along the routes (shown as yellow straight lines) to runway 17R. Once in the Active Movement Area (AMA), the aircraft on a given route (including a queue lane) may not overtake each other, i.e., the relative sequence of aircraft in individual routes are fixed. In Fig. 1, three separate queues are shown for runway 17R. Most of the airports in the US have three or fewer queues at the runways. In some airports, there may also be holding areas near the runway that allow the sequences in individual queues to be changed.

The inputs to the SRS algorithm are the following:

1. Spot, surface route, and departure fix or arrival runway exit assigned to each aircraft; The location of the aircraft on the airport surface imposes precedence constraints on the sequence in which the aircraft can use the runway. These precedence constraints are handled through a queue structure.
2. The weight class and operation type (take-off, landing or crossing) of each aircraft to be scheduled; This is used to determine the minimum separation requirements between pairs of aircraft.
3. Individual time-windows of intended landing for arriving aircraft or take-off times for departing aircraft; This information will be used for handling all arrivals and the subset of departing flights constrained by traffic management initiatives (e.g. flights under Expected Departure Clearance Time (EDCT) restrictions due to a Ground Delay Program or Call For Release (CFR) due to local flow management restriction).

Since most airports have a limited number of departure queues (three or less) and the relative sequence of aircraft in each queue is fixed, one may be incorrectly tempted to assume that obtaining tractable solution times for a three queue problem may be sufficient for practical application of SRS. This is due to several other nuances of the problem. First, the landing and crossing aircraft have their own separate queues. Secondly, the SRS algorithm does not schedule for aircraft in the departure queue and taxiways only, but also for the aircraft that may be at the spots and/or gates. An aircraft at the spot (or gate) may enter the taxiway in front of (or behind) another aircraft at a different spot going to the same queue. Indeed, at many airports the ATCT ground controller will actively manage spot clearance to setup sequences for the local controller who is responsible for aircraft operations at the runway. Hence each spot must be considered as a possible "virtual queue" at the runway and to correctly solve the SRS, the algorithm should be able to obtain computationally tractable solution for the problems involving a large number of queues (in many cases, >10).

III. Single Runway Scheduling Algorithms

We employ three main algorithms for the solution of the SRS problem. The first method is based on a complete enumeration of the solution space and finds the optimal solution to the problem. This method is computationally tractable for small instances of the problem only, and hence we develop two heuristic solution techniques to provide solutions for larger instances. Runway scheduling is a critical component of real-time decision support tools for ATCT controllers, and consequently, both computation time and solution quality are critical factors in deciding a solution technique for SRS.

A. Exact Dynamic Programming (EDP) formulation

Dynamic programming is an algorithmic paradigm in which a problem is solved by identifying a collection of sub-problems and tackling them one by one, smallest first, using the answers to small problems to help figure out larger ones, until the entire problem is solved. In dynamic programming we are not given a graph; the directed acyclic graph is implicit. Its nodes are the sub-problems we define, and its edges are the dependencies between the sub-problems. There are many dynamic programming formulations for various types of runway scheduling.[8-13] The proposed algorithm uses a state definition similar to the one in Ref. [8] and [10] and adds additional information to the state to handle the constraints specific to SRS.

Let A, D, C denote the set of all arrivals, departures and crossing aircraft to be scheduled. Let the total number of aircraft be n, i.e., $|A| + |D| + |C| = n$. Let $\mathbb{A} = A \cup D \cup C$ denote the set of all aircraft. Let $t(i), \forall\, i \in \mathbb{A}$ be the decision variable denoting the runway use time (landing time, take-off time or crossing time) for the i^{th} aircraft. Let $operation(i)$ identify the operation type, i.e., whether the i^{th} aircraft is an arrival, departure or crossing aircraft. Aircraft i can use the runway only after its earliest available time $\alpha(i)$ and should do so before the latest time $\beta(i)$. In our problem, we assume that the landing time cannot be significantly changed, and we allow only small perturbations, δ and constraint the latest times to $\beta(i) = \alpha(i) + \delta, \forall i \in A$. In out simulations, we have used a value

of $\delta = 5$ seconds. Moreover, for crossings and departures without time-window constraints there are no latest time constraints, i.e., $\beta(i) = \infty$.

For departure aircraft, let $head(i)$ denote the heading of an aircraft. For DFW, this takes a value 0 or 1 and is used to plan for divergent heading operations. Let $type(i), \forall i \in D$ denote the weight class type of the i^{th} aircraft. Let there be G distinct weight classes. The weight class of departures allows us to determine the minimum separation requirements between pairs of aircraft. If aircraft i departs before aircraft j, where $i, j \in D$, then they must be temporally separated by $sep_D(type(i), type(j))$ for j to avoid the wake vortex stream from aircraft i. If $head(i) \neq head(j)$ and divergent heading operations are allowed, then aircraft j is allowed a reduced separation given by:

$$sep_D(type(i), type(j)) - rnav(head(i), head(j), type(i), type(j))$$

where $rnav(\dots)$ is the reduction in separation values for divergent heading operations.

For a pair of aircraft i, j, if at least one of them belong to the set C, then the separation between them is given by $sep_C(operation(i), operation(j))$. A departure must wait until the aircraft crossing the runway has cleared completely. Similarly, a crossing aircraft must wait until the departure/arrival has cleared the runway. If both $i, j \in C$, there needs to be a separation between any two consecutive crossings.

For a pair of aircraft i, j, if at least one of them belong to the set A, then the separation between them is given by $sep_A(operation(i), operation(j))$. A departure/crossing must wait until the landing aircraft has cleared the runway completely. Similarly, a landing can occur only if the departure/crossing has cleared the runway. If both $i, j \in A$, there needs to be a separation between the two consecutive arrivals. Since we do not change the landing times (except for small perturbation δ) and the provided initial landing times have the required separation, we can safely assume that this separation is always satisfied.

Since we do not allow the relative sequence of arrivals to change, they can be considered to form a single virtual queue ordered by the earliest available times, with the earliest aircraft at the front of the queue. Even though there are multiple physical crossing queues, the non-dependence of crossings on aircraft types allows for merging the crossing traffic into a single queue. The departures in the active movement area are assigned to the physical queues at the runways. The departures in the ramp area are assigned to virtual queues corresponding to their assigned spots. Departures still at the gates are assigned to virtual queues corresponding to their weight class. Let us suppose that the aircraft can be assigned to Q queues. Let the order of the aircraft in the i^{th} queue be denoted by the sequence $(a_1^i, a_2^i, a_3^i, \dots, a_{k_i}^i)$ where k_i is the size of the i^{th} queue. The queue is ordered with $a_{k_i}^i$ at the front of the queue with $a_1^i > a_2^i > a_3^i, > \cdots > a_{k_i}^i$. This queue structure imposes implicit precedence constraints on the aircraft. Let the initial size of each queue be $q_i, i = 1, \dots, Q$.

A state $S = (h, wc, op, k_1, k_2, k_3, \dots, k_Q)$[§] is defined by the heading of the last departure h, the weight class of the last departure wc, the last operation type op, and the number of aircraft remaining in each queue. Depending on the sequence history, each state may be associated with multiple partial schedules/sequences. We denote a state S with a partial sequence s by S_s. For example, suppose queues 1 and 2 contain departures and the aircraft at the head of the two queues have the same weight class and heading. Consider two possible partial solutions for the algorithm; 's_1' where it picks aircraft from queue 1 followed by queue 2, and 's_2' where the order is reversed. In both these cases, the state corresponding to the two partial solutions is the same state $S' = (h', wc', departure, q_1 - 1, q_2 - 1, q_3, \dots, q_Q)$. In this formulation, each state may contain a large number of solutions.

To check the quality of solutions in each state and to remove inferior partial sequences, a multi-objective cost criteria is used. It consists of three components,

1. $Last(S_s)$: Last time a departure took off in the partial sequence s corresponding to state S.
2. $Span(S_s)$: Makespan corresponding to the last time the runway was used for landing, crossing or take-off in the partial sequence s corresponding to state S.
3. $Delay(S_s)$: Total delay of all the aircraft in the partial sequence s corresponding to state S. It is equivalent to $\sum_{i \in s}(t(i) - \alpha(i))$.

A stage is defined as the set of solutions whose partial sequences are the same size. We initialize our algorithm with an initial state $S_0 = (0, 0, 0, q_1, q_2, \dots, q_Q)$ with the costs initialized to $Delay(S_0) = 0, Span(S_0) = 0$ and $Last(S_0) = -\infty$. We next show how we can recursively progress from one stage to another and discuss how the cost vectors get updated.

[§] If there are no arrival landings, we can use a reduced state $S = (h, wc, k_1, k_2, k_3, \dots, k_Q)$

Suppose we are provided with $S'_{s'} = (h', wc', op', k'_1, k'_2, \dots, k'_Q)$ and an aircraft from queue l uses the runway and the partial solution s' gets updated to a new sequence s. The new sequence $s = (s', a^l_{k_l})$ corresponds to a new state $S_s = (h, wc, op, k_1, k_2, \dots, k_Q)$,
where $op = operation(a^l_{k_l})$ and for each queue $i = 1, \dots, Q$,

$$k_i = \begin{cases} k'_i - 1, & \text{if } i = l, \\ k'_i, & \text{otherwise,} \end{cases}$$

and

$$h = \begin{cases} h' & \text{if } a^l_{k_l} \text{ is not a departure,} \\ head(a^l_{k_l}) & \text{if } a^l_{k_l} \text{ is a departure,} \end{cases}$$

and

$$wc = \begin{cases} wc' & \text{if } a^l_{k_l} \text{ is not a departure,} \\ type(a^l_{k_l}) & \text{if } a^l_{k_l} \text{ is a departure.} \end{cases}$$

Before we provide recursive formulas for the cost vectors, let us define a couple of auxiliary variables:
If $a^l_{k_l}$ is a departure, then let the departure time dep_time be defined as

$$dep\_time = \begin{cases} \max\left(Last(S'_{s'}) + sep_D(wc', wc), Span(S'_{s'}) + sep_C(op', op), \alpha(a^l_{k_l})\right) & if\ op' = crossing, \\ \max\left(Last(S'_{s'}) + sep_D(wc', wc), Span(S'_{s'}) + sep_A(op', op), \alpha(a^l_{k_l})\right) & if\ op' = arrival, \\ \max\left(Last(S'_{s'}) + sep_D(wc', wc) - rnav(h', h, wc', wc), \alpha(a^l_{k_l})\right) & if\ h \neq h, \\ \max\left(Last(S'_{s'}) + sep_D(wc', wc), \alpha(a^l_{k_l})\right) & otherwise, \end{cases}$$

and if $dep\_time > \beta(a^l_{k_l})$, then set $dep\_time = \infty$.
If $a^l_{k_l}$ is an arrival, then let the landing time arr_time be defined as

$$arr\_time = \max\left(Span(S'_{s'}) + sep_A(op', op), \alpha(a^l_{k_l})\right)$$

and if $arr\_time > \beta(a^l_{k_l})$, then set $arr\_time = \infty$.

With the two new variables defined above, it becomes easier to write the recursion for the cost function values. We can calculate the last departure time using the following recursion,

$$Last(S_s) = \begin{cases} Last(S'_{s'}) & if\ op \neq departure, \\ dep\_time & otherwise. \end{cases}$$

Furthermore, we can calculate the makespan using

$$Span(S_s) = \begin{cases} \max\left(Span(S'_{s'}) + sep_C(op', op), \alpha(a^l_{k_l})\right) & if\ op = crossing \\ Last(S_s) & if\ op = departure \\ arr\_time & if\ op = arrival \end{cases}$$

Delay is calculated as

$$Delay(S_s) = Delay(S'_{s'}) + Span(S_s) - \alpha(a^l_{k_l}).$$

Elimination Step: Consider two partial solutions s and s' corresponding to the same state S. If $Delay(S_s) \leq Delay(S'_{s'})$, $Span(S_s) \leq Span(S'_{s'})$ and $Last(S_s) \leq Last(S'_{s'})$ and at least one of the three is a strict inequality, then we say that s dominates s'. The partial solution s' can be removed from the possible solutions. A partial solution s in state S is also removed if any of the cost values is infinite.

Starting from the initial solution S_0 at stage 0, we recursively go through *n* stages of the DP. In the final stage we have states of the form $S^n = (h, wc, op, 0, 0, 0, \dots, 0)$ with all queues being empty. Among all the pareto-optimal solutions in the last stage, we choose the solution with the least delay cost as our preferred solution.

B. Restricted Dynamic Program (RDP) heuristics formulation

The EDP algorithm described in the previous section is analogous to a breadth first search of the solution space. The number of nodes in the solution space is bounded by $2^{G+4}\prod_{i=1}^{q}(k_i + 1)$. The elimination step of the EDP algorithm eliminates those nodes that are dominated by other nodes. Even with the elimination of a significant number of states, some stages of the exact DP formulation of the SRS could have a large number of states for moderately sized scenarios. To avoid the large number of states of the EDP, a restricted DP heuristic algorithm is applied. This heuristic was first proposed for the Traveling Salesman Problem.[22] In each stage of the Restricted

Dynamic Program (RDP), only a restricted subset of H states with the smallest delay is kept. Increasing the value of H should yield better solutions, but will also result in higher computation times. We deploy three variants of the algorithm with the value of H set to 10,000, 20,000, and 30,000, respectively.

C. Insertion and Local Search (ILS) heuristics

In this section, a heuristic with fast computation time that produces good quality solutions for the SRS is explained. This heuristic was used in the human-in-the-loop evaluation of SARDA concept in 2012.[21] The heuristic starts with a feasible First Come First Served (FCFS) sequence of runway use. The heuristic then progresses along this initial sequence, in the i^{th} iteration fixes the first i aircraft in the sequence, and then does a local neighborhood search on the remainder of the sequence. It iteratively inserts the *best* aircraft from the non-fixed part to the fixed part of the sequence. Based on the heuristics employed, we call this algorithm the Insertion and Local Search (ILS) algorithm.

A few notations are presented before discussing the ILS algorithm. Consider a feasible sequence $H = (1, 2, 3, \dots, n)$ of n aircraft. Let $t(i)$ be the runway use time of the i^{th} aircraft in the sequence; $t(i)$ can be calculated in polynomial time. The runway use time is the earliest time that satisfies all the time-window and separation constraints and constitutes a solution to SRS. Let $Delay(H) = \sum_{i \in H}(t(i) - \alpha(i))$ and $Span(H) = max_{i \in H} t(i)$ be the objective values corresponding to this solution. A sequence H_1 is preferred over a sequence H_2

- if $Delay(H_1) < Delay(H_2)$,
- or $Delay(H_1) = Delay(H_2)$ and $Span(H_1) < Span(H_2)$.

This allows for comparison of two runway sequences and chooses the one that reduces overall delay, or in cases with identical delay selects the one with lower makespan. The preference of H_1 over H_2 is represented concisely as $H_1 \prec H_2$.

Let H^{i+1} denote a feasible sequence of n aircraft comprised of two subsequences (H_{fixed}, H_{free}) of size i and $(n - i)$, respectively.

Given a sequence $H^{i+1} = (H_{fixed}, H_{free})$, a subsequence H_{k_free} is said to be in the neighborhood of H_{fixed} if (1) the sequence (H_{fixed}, H_{k_free}) is feasible and, (2) H_{k_free} can be constructed using a permutation of the first k aircraft of H_{free} with the other aircraft retaining its position. Let the neighborhood be denoted by $\mathcal{N}(H_{fixed})$.

The heuristic starts with a feasible First Come First Served (FCFS) sequence of runway use. If no time-window constraints are considered, then the FCFS sequence is easy to construct. It is the ordered list of aircraft sorted by their earliest available times. Addition of landing aircraft and/or departures with time-window restrictions can make the computation of the FCFS sequence tricky, since sorting by the earliest available times may give rise to an infeasible sequence. In this case, a FCFS solution is generated by first considering the time-window constrained aircraft and assigning them a runway use time. The other aircraft are then sorted and sequentially inserted, in ascending order, into the solution to the first available slots while ensuring that it does not cause any conflicts with the aircraft previously considered in the solution. This modified FCFS solution is then used as the initial solution to the algorithm.

ILS Algorithm:

1. The feasible sequence obtained from a FCFS algorithm is used as the initial sequence H^1. Initialize parameter k to define the neighborhood of a sequence. Initialize num_swaps to a non-zero integer (=1).
2. Do the following as long as num_swaps is greater than zero.
   - Set num_swaps =0
   - For i ranging from 1 to $n - k$
     - Set $H^{i+1} = H^i$
     - Let $H^i = (H_{fixed}, H_{free})$
     - Construct the k-swap neighborhood $\mathcal{N}(H_{fixed})$
     - For each subsequence $F \in \mathcal{N}(H_{fixed})$
       - Set $H_F^i = (H_{fixed}, F)$
       - If $H_F^i \prec H^{i+1}$
         - Set $H^{i+1} = H_F^i$
         - Increment num_swaps by 1

The algorithm finds a local minima corresponding to the initial sequence and the parameter k.

IV. Results

In the previous section, three algorithms were provided for the SRS problem. Runway scheduling is an important component of any surface decision support tool and requires an algorithm that produces good quality solutions quickly. In this section, we conduct a comparative study of the three algorithms to determine the suitability of the algorithms for a real-time decision support tool. The algorithms are referred to as EDP (the exact/optimal solution), RDP10K (restricted dynamic programming with 10,000 states), RDP20K (restricted dynamic programming with 20,000 states), RDP30K (restricted dynamic programming with 30000 states), and ILS (insertion and local search heuristics). We have used a value of $k = 7$, for the ILS algorithms.

In our simulations, we considered only the east side of the DFW airport operating in South Flow configuration (see Fig. 1). Since runway 17R is not used for arrival landings, we considered only departures and runway crossings. For this study, a planning window of 15 minutes was selected. Each scenario has 20 departures and 15 arrivals, and represents a heavy traffic condition. The earliest available times (α_i) were uniformly distributed within 0-900 seconds. The fleet mix distributions were chosen to represent traffic at DFW, with 80% of weight-class Large, 10% of class Heavy and 10% of B75x. The wake vortex separation matrix, given in Table 1, was used for the simulations. The columns represent the weight class for leading aircraft whereas the rows represent the following aircraft. For example, if a large aircraft follows a heavy aircraft, then they must be separated by 90 seconds. The departures were randomly assigned a heading value of 0 or 1. Non-heavy departures to different headings require a reduced separation as part of divergent heading operations. This reduction was empirically observed to be around 6 seconds. For example, two departures of weight class large to divergent heading require a separation of 39 seconds. Arrivals can cross runway 17R 40 seconds after a departure takes off and take 21 seconds to clear the runway. If two arrivals cross the runway consecutively, the temporal separation between them is 6 seconds.

Table 1. Wake vortex separation (in seconds) for departure aircraft.

	Heavy	Large	B75x
Heavy	70	90	90
Large	45	45	45
B75x	70	70	70

Although there are four separate queues for runway crossing, they can be merged into a single virtual queue. There are also three physical departure queues at runway 17R. The aircraft in each queue have implicit precedence constraints imposed on them. Since we are looking at a 15-minute planning window, some of the departure aircraft could still be at the spots or gates. Two aircraft at different spots do not have a precedence constraint even if they are going to the same physical queue at the runway. Anecdotal evidence also suggests that the ATCT ground controller tries to setup an efficient sequence for the local controller. Hence each spot is considered as a possible "virtual queue" at the runway. To correctly solve the SRS, the algorithm should be able to obtain computationally tractable solutions for problems involving a large number of queues. To study the effect of increasing the number of queues on the algorithms, scenarios were generated with varying numbers of queues (from 3 to 10) with hundred different scenarios generated for each queue number. The arrivals (crossing) were assigned to a single queue, and the departures were randomly assigned to the other queues.

A. Computation times

Figures 2 and 3 show the computation time of the different algorithms with respect to the number of queues. Each point shows the average time (in msec) for an algorithm to obtain the solution for the 100 random scenarios. The graph shows that the EDP provides fast solutions for small queue sizes (<6 queues), but grows exponentially as the number of queues increases. The variations in computation times of the different heuristic algorithms are also shown in Figure 3. The RDP solution time increases with increasing H (the number of states kept in each stage of the DP). The ILS algorithm has a nearly constant solution time irrespective of the number of queues. Figure 3 also shows that the RDP10K heuristics starts restricting the search space at a queue number of 6, RDP20K at a queue number of 7 and RDP 30K at a queue number of 8. Based on the RDP algorithm, the increase in computation time after this point is expected to be linear and the data shows the same.

In NASA's human-in-the-loop simulation, the runway schedule was calculated every 10 seconds. For large queue numbers we find that ILS and RDP10K are the two algorithms that meet the computation time requirement of providing a solution within 10 seconds.

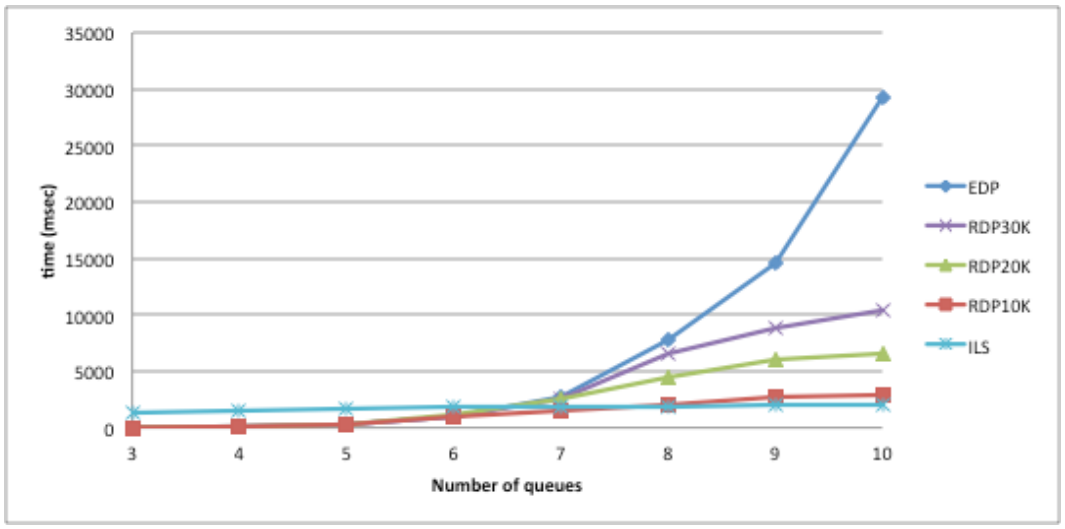

Figure 2. Computation time for the algorithms. Each point denotes the average value over 100 different scenarios.

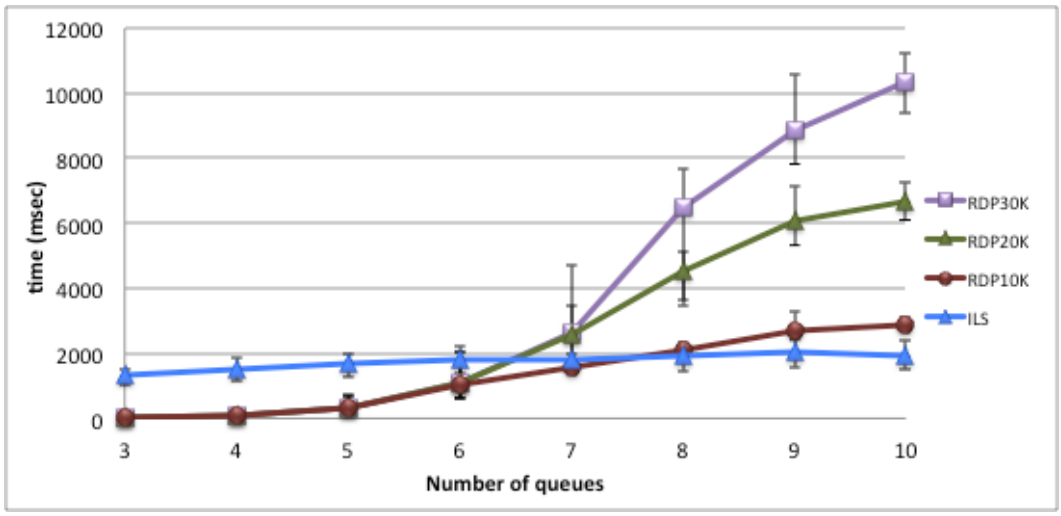

Figure 3. Computation time for the heuristic algorithms. Each point denotes the average value over 100 different scenarios, and the whiskers represent 10th and 90th percentile.

B. Quality of heuristic solution

The solutions of the heuristic algorithms (RDP and ILS) are compared to the optimal solution from the EDP algorithm. For each solution, the total delay of all the aircraft are computed and the percentage difference from the optimal solutions is calculated. Reference [5] shows that optimizing for total delay results in small deviations from the optimal throughput, whereas optimizing for throughput results in large deviations in total delay. For this reason, total delay was chosen as the objective for the scheduler.

These results are plotted in Fig. 4. For each queue number, the algorithms were applied to 100 scenario instances, and each point represents the average percentage difference in total delay from EDP's delay values for the respective algorithm. The whiskers represent the 10th and 90th percentiles. On average the RDP solutions get worse with increasing queue number, whereas the ILS algorithm consistently produces solutions that are within 10% of the optimal. From this graph, it is evident that for large queue sizes (>6), ILS solutions are of much higher quality than the others.

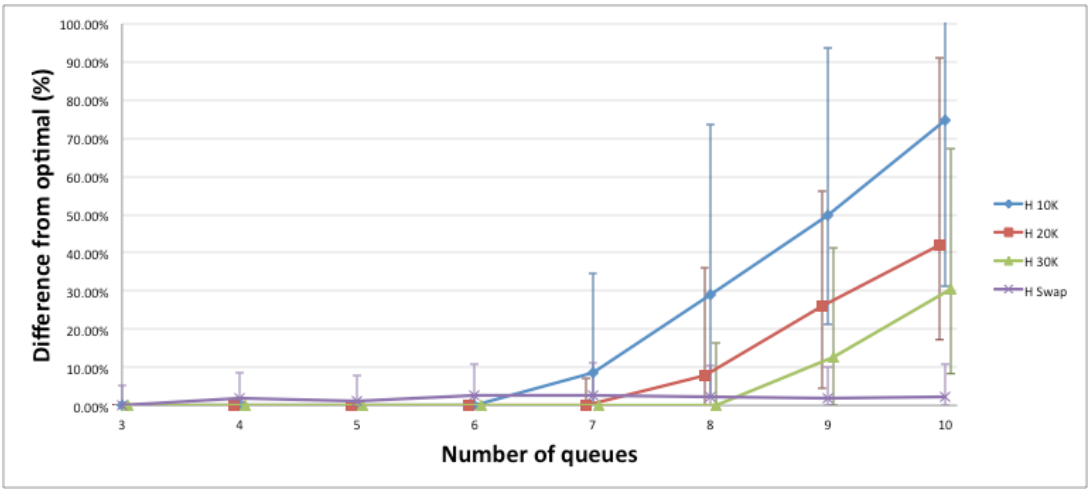

Figure 4. The percent difference of total delay from the optimal delay as a function of queue number.

V. Conclusion

In this paper, we formulated three algorithms for Single Runway Scheduling of airport surface traffic: Exact Dynamic Programming (EDP); Restricted Dynamic Programming (RDP); and Insertion and Local Search (ILS). A comparative study of the three algorithms was conducted by performing simulations for the east side of the Dallas/Fort Worth International Airport (DFW). For the cases with more than six queues the ILS heuristics performed significantly better than the other algorithms and produced good quality solutions in a relatively short computation time. Based on the presented results, we can conclude that among the three algorithms, the ILS heuristics is the most suitable candidate for application in tactical surface decision support tools.

References

[1]Idris, H., Delcaire, B., Anagnostakis, I., Hall, W. D., Pujet, N., Feron, E., et al., "Identification of Flow Constraint and Control Points in Departure Operations at Airport Systems," AIAA Guidance, Navigation, and Control Conference and Exhibit, Boston, MA, Aug. 10-12, 1998.

[2]Bianco, L., Rinaldi, G., Sassano, A., delle Ricerche, C.N. and Manzoni, V., "A Combinatorial Optimization Approach to Aircraft Sequencing Problem," Flow Control of Congested Networks, edited by A. R. Odoni, L. Bianco, and G. Szego¨, NATO ASI Series, Series F: Computer and Systems Science 38, 323–339, Springer-Verlag, Berlin, 1987.

[3]Bianco, L., Ricciardelli, S., Rinaldi, G. and Sassano, A., "Scheduling Tasks with Sequence-Dependent Processing Times," Naval Research Logistics, Volume 35, Number 2, 177–184, 1988.

[4]Bianco, L. and Bielli, M., "System Aspects and Optimization Models in ATC Planning," Large Scale Computation and Information Processing in Air Traffic Control, edited by L. Bianco and A. R. Odoni, 47–99, Springer-Verlag, Berlin, 1993.

[5]Gupta, G., Malik, W., and Jung, Y. C., "A Mixed Integer Linear Program for Airport Departure Scheduling," 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO), AIAA, Hilton Head, South Carolina, 2009.

[6]Gupta, G., Malik, W., and Jung, Y. C., "Incorporating Active Runway Crossings in Airport Departure Scheduling," AIAA Guidance, Navigation, and Control (GNC) Conference, Toronto, Canada, 2010.

[7]Brinton, C. R., "An Implicit Enumeration Algorithm for Arrival Aircraft Scheduling," In Proceedings of the IEEE/AIAA 11th Digital Avionics Systems Conference, Seattle, WA, 1992.

[8]Psaraftis, H. N., "A Dynamic Programming Approach for Sequencing Groups of Identical Jobs," Operations Research, pp. 1347-1359, Vol. 28, No. 6, 1980.

[9]Trivizas, D. A., "Optimal Scheduling with Maximum Position Shift (MPS) Constraints: A Runway Scheduling Application," Journal of Navigation, 51:250–266, 1998.

[10]Balakrishnan, H., & Chandran, B., "Scheduling Aircraft Landings Under Constrained Position Shifting," AIAA Guidance, Navigation and Control Conference and Exhibit, Keystone, CO, 2006.

[11]Balakrishnan, H., and Chandran, B., "Efficient and Equitable Departure Scheduling in Real-time: New Approaches to Old Problems," 7th USA - Europe Air Traffic Management Research and Development Seminar, Barcelona, Spain, 2007.

[12]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 (GNC) Conference, Chicago, IL, 2009.

[13]Montoya, J., Rathinam, S. and Wood, Z., "Multiobjective Departure Runway Scheduling Using Dynamic Programming," IEEE Transactions on Intelligent Transportation Systems, 15(1), pp.399-413, 2014.

[14]Dear, R. G., and Sherif, Y. S., "Dynamic Scheduling of Aircraft in High Density Terminal Areas," Microelectronics Reliability, pp. 743-749, Vol. 29, No. 5, 1989.
[15]Bianco, L., Dell'Olmo, P., and Giordani, S., "Minimizing Total Completion Time Subject to Release Dates and Sequence-Dependent Processing Times," Annals of Operations Research, 86:393–415, 1999.
[16]Stiverson, A. W., "A Study of Heuristic Approaches for Runway Scheduling for the Dallas-Fort Worth Airport," Master's thesis, Texas A&M University, USA, 2009.
[17]Atkin, J.A.D., Burke, E.K., Greenwood. J.S., and Reeson, D., "A metaheuristic approach to aircraft departure scheduling at London Heathrow airport," 9th International Conference on Computer-Aided Scheduling of Public Transport, San Diego, CA, USA, 2004.
[18]Atkin, J. A. D., Burke, E. K., Greenwood, J. S., and Reeson, D., "An Examination of Take-Off Scheduling Constraints at London Heathrow Airport," In Electronic proceedings of the 10th International Conference on Computer-Aided Scheduling of Public Transport, 2006.
[19]Bennell, J. A., Mesgarpour, M., and Potts, C. N., "Airport runway scheduling," Annals of Operations Research, 204(1), 249-270, 2013.
[20]Jung, Y., Hoang, T., Montoya, J., Gupta, G., Malik, W., Tobias, L., and Wang, H., "Performance Evaluation of a Surface Traffic Management Tool for Dallas/Fort Worth International Airport," the Ninth USA/Europe Air Traffic Management Research and Development Seminars, Berlin, Germany, June 2011.
[21]Gupta, G., Malik, W., Tobias, L., Jung, Y., Hoang, T., and Hayashi, M., "Performance Evaluation of Individual Aircraft Based Advisory Concept for Surface Management," the Tenth USA/Europe Air Traffic Management Research and Development Seminars, Chicago, IL, June 2013.
[22]Malandraki C., and Dial, R.B., "A restricted dynamic programming heuristic algorithm for the time dependent traveling salesman problem," European Journal of Operational Research, Vol. 90, pp. 45–55, 1996.