AIAA 2013-4275

Analysis of Airport Surface Schedulers Using Fast-time Simulation

Justin Montoya* and Robert Windhorst†
NASA Ames Research Center, Moffet Field, CA, 94035

Zhifan Zhu‡ and Sergei Gridnev§
Stinger Ghaffarian Technologies Inc., Moffett Field, CA, 94035

Katy J. Griffin¶, Aditya Saraf‖, and Steve Stroiney**
Saab Sensis Corporation, Campbell, CA, 95008

Using a fast-time simulation tool Surface Operations Simulator and Scheduler, three gate pushback metering concepts are tested on a mock-up at Charlotte Douglas International Airport. The three concepts include (1) a mixed-integer-linear program to minimize aircraft delay, (2) a heuristic scheduler which uses aggregate aircraft counts to meter traffic, and (3) a first-come-first-served method that releases aircraft as soon as they are ready. A realistic scenario is created using surveillance data. By perturbing departure pushback times and arrival on-times, approximately 500 new scenarios are created from the original scenario. Additionally, pushback duration uncertainty and the inclusion of a miles-in-trail restriction are tested as independent variables. Results show that the calibration of certain model parameters on the mixed-integer-linear-program and heuristic scheduler strongly affect aircraft taxi times and delays. Also, the mixed-integer-linear-program is more robust to pushback duration uncertainties when there is a miles-in-trail restriction present.

I. Introduction

The process of pushing departure aircraft back from gates is accomplished, nominally, on a first-available-first-served basis. First-available-first-served operates by releasing departure aircraft from gates after the pilot has communicated to a ramp controller that they are ready for pushback. Unfortunately when departure aircraft are given clearance to pushback, the current congestion in the active movement area is not considered. This means that most pushbacks are uncoordinated resulting in a line of departure aircraft burning fuel and waiting for their turn to use the departure runway [1].

With advances in surveillance technology at airports [2], coordinating departure push back procedures based on departure runway demand is feasible. Authors in [3] describe a surface congestion management tool at John F. Kennedy International Airport that holds back departure aircraft at gates and releases them at specific times based on their position in a virtual queue and the current demand at the departure runway. Furthermore, field studies at Boston Logan International Airport tested a system that uses pushback rates to restrict the flow of aircraft pushing back from gates [4]. Finally, authors in [5] used a high fidelity human-in-the-loop simulation of Dallas/Fort Worth International Airport to test an advisory system that finds delay-optimal gate release times.

*Aerospace Engineer, NASA Ames Research Center, Moffett Field, CA 94035. AIAA Member.
†Aerospace Engineer, NASA Ames Research Center, Moffett Field, CA 94035. AIAA Senior Member.
‡Software Engineer, Stinger Ghaffarian Technologies Inc., Moffett Field, CA, 94035.
§Software Engineer, Stinger Ghaffarian Technologies Inc., Moffett Field, CA, 94035.
¶Research Engineer, Advanced Development Saab Sensis Corporation, AIAA Member
‖Senior Research Engineer, Advanced Development Saab Sensis Corporation, AIAA Member
**Senior Research Engineer, Advanced Development Saab Sensis Corporation, AIAA Member

The advisory tools described in [4] are different from those described in [3] and [5]. The advisory tools described in [4] *do not* use an optimal sequence for aircraft to use the runway when making a decision on how long aircraft should wait at gates. Instead, these tools constrain the number of departure aircraft actively taxiing to runways, without starving runways of departure aircraft. In contrast, advisory tools like the Spot and Runway Departure Advisor (SARDA) [5], use a delay-optimal sequence to calculate how long aircraft should hold at gates. Accounting for taxi times and traffic, SARDA provides pushback clearance times for aircraft at gates based on the prescribed optimal runway schedule. Controllers may or may not control flights in the ramps and active movement area to achieve the order prescribed at the runway.

Because of the uncertainty in aircraft pushback and taxiing, optimal runway sequencing may not help lower average aircraft delays. In fact, simpler advisory tools, which do not calculate optimal runway sequences, may provide better or equal results. Since the number of possible scenarios at airports varies dramatically, assessing the quality of a scheduler is difficult. From a scheduler's point of view this means that the size of the input space is large. Recently, fast-time simulation tools are available for assessing scheduler performance under various operating conditions and scenarios.

In this paper, NASA's Surface Operation Simulator and Scheduler (SOSS) [6][7] is used to analyze the performance of three schedulers for managing surface traffic. Hundreds of simulations are assessed under varying degrees of uncertainty and operational constraints. Also, unlike other previous work, these schedulers will be tested over a realistic two hour scenario at Charlotte Douglas International Airport, allowing for schedulers to be tested under dynamic conditions where aircraft are constantly leaving and entering successive scheduler calls. Fundamentally, this paper attempts to reveal the capabilities provided by NASA's airport surface fast-time simulation tool SOSS, and to do a meaningful analysis of existing scheduling concepts for managing departure pushback events.

A concept that calculates delay-optimal runway sequences is compared to a concept that uses a heuristic algorithm. The optimal concept is similar to that proposed in [5], while the heuristic algorithm is closer to those described in [4]. Additionally, both concepts are compared against a baseline concept that uses a first-come-first-served algorithm. Furthermore using a realistic scenario at Charlotte Douglas International Airport, thousands of simulations are conducted under uncertainty and operational constraints.

Section II provides a description of the relevant algorithmic strategies for metering departures at the gate. Section III describes simulation tool Surface Operation Simulator and Scheduler (SOSS). Section IV describes the experiment and pertinent independent variables. Section V explores a set of results drawn from the experiments. Finally, section VI provides conclusions.

II. Description of Three Surface Advisory Algorithms

This section describes how each of the algorithmic strategies being studied functions. First a description of a possible SARDA scheduler is given. Next, a heuristic algorithm based on the work done in [4] is given. Finally, a baseline algorithm is described.

A. SARDA Based Scheduling Concept

The core functionality of SARDA is a scheduling concept that consists of two scheduling components. The first component is called the Runway Scheduler (RS) and the second is called the Spot Release Planner (SRP).

First, a formal description of the problem solved by the RS is given. Given a set of aircraft, A, time separation parameters, $s_{i,j}$, for each aircraft $i \neq j \in A$, the earliest time, e_i, and latest time, l_i, each $i \in A$ can use the runway, and a set P_i of aircraft who must use the runway before $i \in A$, find the set of scheduled times, t_i, that each aircraft should begin to use the runway such that the total delay is minimized. As a

mixed-integer non-linear program this can be represented in the following manner:

Runway **S**cheduling **P**roblem

$$\underset{\{t_i:i\in A\}}{\text{minimize}} \sum_{i\in A} t_i - e_i \tag{1}$$

Time window constraints:

$$e_i \le t_i \le l_i \qquad \forall i \in A, \tag{2}$$

Minimum separation constraints:

$$0 \le (t_j - t_i - s_{i,j}) z_{i,j} \qquad \forall i \ne j \in A, \tag{3}$$

Precedence constraints:

$$t_j \le t_i \qquad \forall i \in A, j \in P_i, \tag{4}$$

Necessary constraints:

$$z_{i,j} + z_{j,i} = 1 \qquad \forall i \ne j \in A, \tag{5}$$

$$z_{i,j} \in \{0,1\} \qquad \forall i,j \in A, \tag{6}$$

$$t_i \in \Re^+ \qquad \forall i \in A. \tag{7}$$

The above model has two sets of variables $z_{i,j}$ and t_j, where $z_{i,j} = 1$ can be interpreted as "aircraft i uses the runway before aircraft j" and $z_{i,j} = 0$ means that "aircraft i uses the runway after aircraft j". The times for aircraft to begin to use the runway are given by t_i. These are calculated by observing separation constraints (equation (3)), precedence constraints (equation (4)), and earliest and latest constraints (equation (2)). The only non-linear equation is equation (3), which can be cast as a linear equation, as shown in [8], and therefore this model can be cast as a Mixed Integer Linear Program (MILP). For a detailed discussion of this model, please see [8], [9], and [10].

In this paper, RS schedules departure and crossing aircraft. Arrival landing times are inputs that block the runway at certain periods of time and cannot be altered. After RS has calculated a schedule for departures to depart and arrivals to cross the runway, SRP calculates times that departure aircraft should leave other locations on the airport surface based upon the RS schedule. In this paper, SRP calculates a pushback time for departures. This time is calculated to meet the originally planned departure time.

Authors in [10] suggest that SRP can simply take each t_i, provided by RS, and subtract off the nominal taxi time from gate to runway. The resulting time calculated would be the time the aircraft is released from its gate.

While simple enough, a previous study [11] found that it is important to include a parameter W_r which controls the amount of time an aircraft is expected to wait at runway r for wheels-off. The insight here is that the nominal taxi time estimate is hardly perfect and perturbed by additional traffic, pilot reaction, controller interactions, and various other operational norms. The side-effect of not including W_r is that the runway could go unused for a period of time. To maintain pressure on runways and manage uncertainty, the authors in this paper adopt a similar strategy by calculating SRP pushback times in the following manner:

$$delay_i = \min\{W_r, t_i - e_i\} \tag{8}$$

$$t_i^{desired} = t_i - delay_i \tag{9}$$

$$PBT_i^{act} = PBT_i^{sched} + (t_i^{desired} - e_i) \tag{10}$$

Equation (8) provides the amount of delay to apply to any departure aircraft i waiting at its gate. The minimum function is used to prevent any aircraft from pushing back from its gate earlier than it possibly can. Equation (9) is the updated wheels-off time for aircraft i (possibly earlier than t_i). Equation (10) then calculates the actual (or advised) pushback time PBT_i^{act} based on the scheduled pushback time PBT_i^{sched}, the desired departure time $t_i^{desired}$, and the earliest time aircraft i can use the runway.

In summary, the variation[a] of SARDA deployed in this paper first calculates a set of t_i values, which are then translated to actual pushback times PBT_i^{act} for each departure aircraft. Fig. 1 summarizes the flow of information:

[a]There could be many implementations developed from the SARDA framework. The key invariable is that SARDA uses individual aircraft parameters to meter and sequence aircraft.

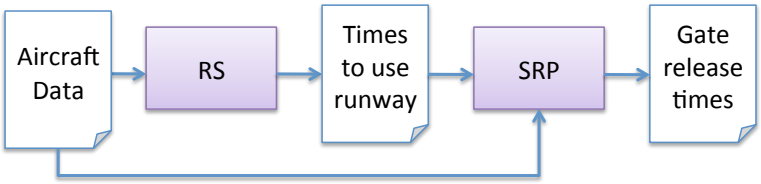

Figure 1: Flow of SARDA scheduler.

B. Heuristic Scheduling Concept

A simple gate holding algorithm is described. For a given runway r, this algorithm works in the following manner:

1. Update to current time t.
2. At time t count the number of departure aircraft pushed back from gate and traveling to runway r. Call this n_r^t.
3. If n_r^t is equal to N_r (i.e., a threshold that is assumed to be greater than 0), then proceed to step 4, otherwise return to step 1.
4. Determine the time, t^r, that the next departure aircraft will ***use*** r amongst those already pushed back from the gate.
5. Amongst the aircraft still waiting at their gates at t, let i^* be the aircraft with the earliest available time to pushback. This is the aircraft that has requested pushback soonest prior to t. If no such aircraft exists at time t, return to step 1.
6. Let aircraft i^* pushback at t^r, and for all other aircraft waiting at gates, set their pushback times to ∞.

The above algorithm keeps the number of departure aircraft traveling to a particular runway (e.g., r) to a prescribed maximum N_r. When N_r aircraft are traveling to the runway, then all other aircraft must wait until a departure uses the runway. The moment a departure uses the runway, the aircraft who requested pushback first is allowed to pushback.

This algorithm is similar, but not identical to the concept presented in [4]. The concept in [4] operates by accounting for individual aircraft taxi-times and allowing controllers to assess when departure aircraft should pushback based on a rate constraint (e.g., 10 aircraft in 15 minutes). In contrast, this simple gate holding concept does not directly account for aircraft taxi times nor does it advise a rate constraint. This algorithm is similar in that it tries to maintain a maximum number of aircraft using a given runway.

C. Baseline Scheduling Concept

The baseline gate metering concept operates on a first-available-first-served process by allowing aircraft to pushback from their gates as soon as possible. This means that any aircraft who has requested pushback can pushback immediately, without any consideration of down stream affects.

III. Surface Operations Simulator and Scheduler (SOSS)

A. Node-link Network Description

Using airport adaptation data, SOSS models various airport geometries. A node-link model of airports is provided by SOSS representing taxi intersections throughout the ramp, active-movement area, and up to the runway. Fig. 2 shows the node-link model used for Charlotte Douglas International Airport (CLT). Shown on the left of Fig. 2 is a full image of the node-link network of CLT, and provided on the right is a blown-up image indicating specific node types. Starting from the top left, SOSS provides departure, queue, taxi, crossing, spot, arrival, and ramp node types. These node types also have corresponding link types.

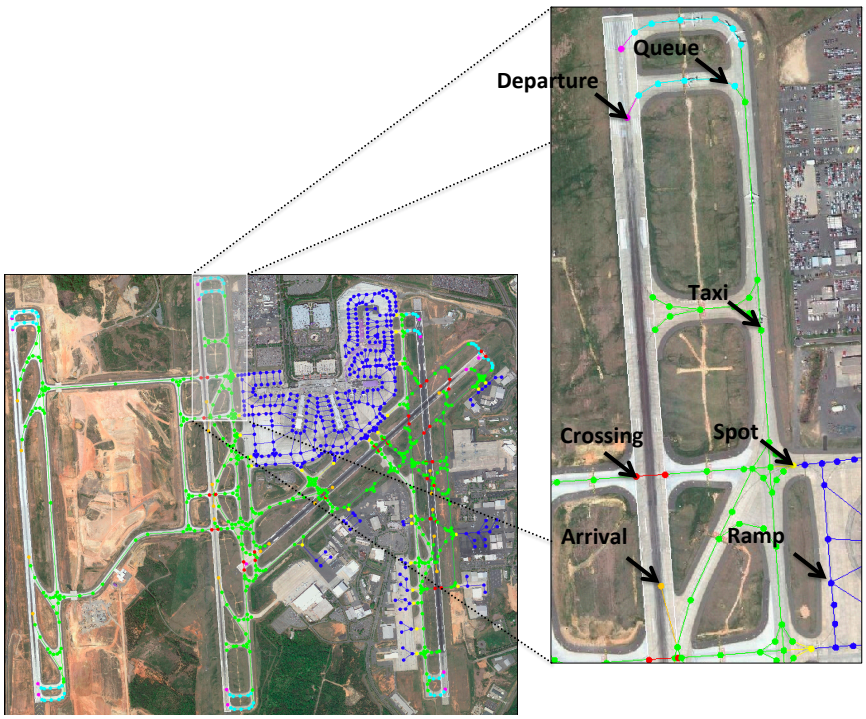

Figure 2: A node-link network of Charlotte Douglas International Airport.

A departure node/link indicates the area where departure aircraft begin their procedure to depart. Just behind the departure node/link is a collection of nodes/links that indicate the departure queue area. The queue area is primarily where departures queue up for departure. Another set of links/nodes are the taxi nodes, which correspond to the active movement area. Also indicated on this map are crossing nodes, where aircraft can begin to cross runways. Spot nodes/links are locations that indicate the border between the ramp area and the active movement area. Next, a single node/link is used for the arrivals, which corresponds to the runway exit location for a landing arrival aircraft. Finally, the ramp nodes/links are home to gates and various intersections throughout the ramp.

B. Movement

SOSS simulates the movement of traffic using a kinematics model that assumes constant-linear acceleration where each aircraft has a prescribed max speed based on the aircraft's current state and type. For example Fig. 3 shows the speed profile of a departure traveling from its gate to runway 18C. This departure aircraft has five main states that it transitions through. When the aircraft is in the ramp, it goes through two of these states, namely, pushback and ramp taxi. The first portion of the speed profile shows a target speed of 4 kts (knots) indicating it is in pushback. After pushback, the departure aircraft accelerates (at a prescribed max) to a target ramp speed of 10 kts. When the aircraft crosses a spot, it moves into the active movement area and is able to taxi at 15 kts. Finally after taxiing, the departure enters the departure queue and must slow down to 10 kts again. Once the departure finally arrives to its final node (e.g, the departure node), it can commence the take-off phase, where accelerations are much higher.

In general, when an aircraft *must* speed up or slow down, it will do so using the max acceleration permitted. Once the aircraft has reached a stop or the target speed, then the aircraft stops accelerating. This process is fairly identical for all aircraft, even for arrivals which have their state transitions in reverse order.

C. Conflict Detection and Resolution

In addition to aircraft movement, SOSS provides a conflict detection and resolution module. The conflict detection and resolution module is decomposed into two separate components: (1) A taxi component and

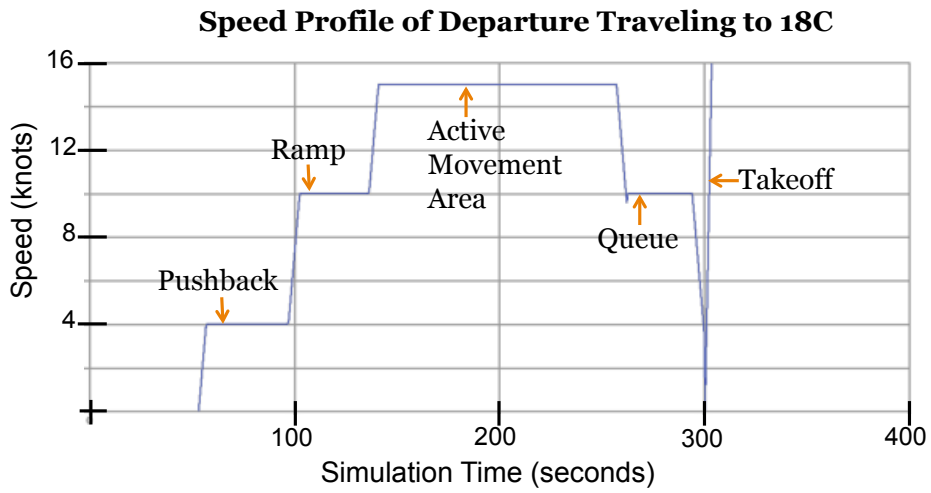

Figure 3: SOSS speed profile for a departure aircraft traveling to runway 18C from its gate.

(2) a runway component.

1. *Taxi Conflict Detection and Resolution*

SOSS maintains a minimum separation between any two taxiing aircraft. SOSS can maintain both lateral and longitudinal separations, but this paper only describes the longitudinal separation since it was the one used.

Any two taxiing aircraft are separated according to speed and a constant separation factor. Separation, in terms of distance, between two aircraft is handled using the following linear model:

$$Taxi\_Separation = v * \beta + \alpha, \tag{11}$$

where v is the speed of the aircraft that is traveling behind another aircraft, β is a positive constant multiplier, and α is a positive constant factor. The idea here is that when the trailing aircraft is traveling slower, it can have a smaller separation between it and the leading aircraft. However, when $v = 0$ then α will safely keep the aircraft separated. If it helps, one can think of these two separation constants as the dynamic (β) and static (α) components.

2. *Runway Conflict Detection and Resolution*

Runway separation rules involve separation between successive departures, arrivals, or crossing traffic at runways. These rules also include intersecting runway separations. To explain these rules, consider Fig. 4. In Fig. 4 the operations performed on CLT's 18C runway and those operations that affect operations on runway 18C are depicted. For instance, a departure heading to a departure fix (e.g., JACAL) must ensure that there is no intersecting traffic from runway 23, no arrivals landing, no runway crossing activity, and must be sufficiently separated from any departure ahead of it. Rules between successive departures are dependent on their weight class and the departure fix they are approaching.

All operations in SOSS are separated in time, and for 18C table 1 gives all separation values. Table 1 contains most inter-aircraft separations (e.g., $s_{i,j}$) used for runway 18C. Table 1a shows the inter-separation between departure aircraft, where column departure aircraft are assumed to lead row departure aircraft for proper separation. For example, if a heavy departure aircraft uses runway 18C prior to a small departure aircraft, then the small aircraft must wait 120 seconds (s) before it can depart. Similarly tables 1b and 1c contain the separation values used between arrivals and departures on runway 18C. Again, these matrices assume that column aircraft lead row aircraft for correct separation.

In addition to the separations in table 1, separation between runway 23 and runway 18C (see Fig. 4) is enforced. Any aircraft that departs or arrives on 18C must wait a minimum of 5-7 seconds as soon as the

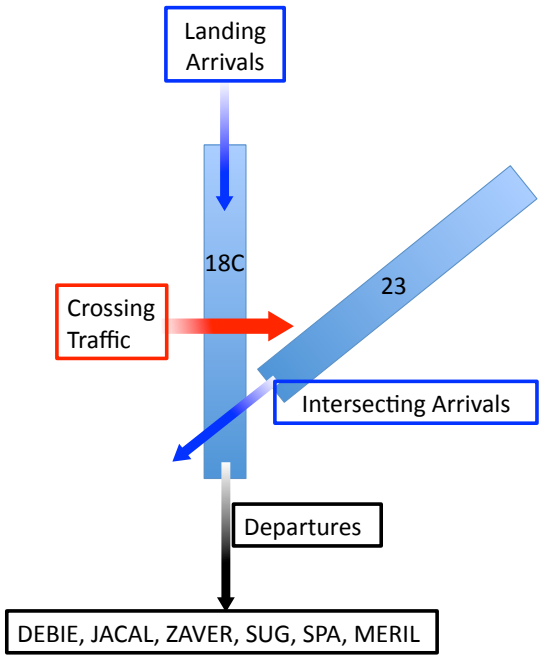

Figure 4: A notional image of runways 18C and 23 at CLT. Arrows indicate various operations occurring relative to runway 18C. At the end of runway 18C there is a collection of immediate departure fixes used by departing aircraft.

Table 1: Separation tables for runway 18C at CLT. All units are in seconds.

(a) Departure (row) vs. Departure (col)

	B75*	Heavy	Large	Small
B75*	90	90	60	60
Heavy	90	90	60	60
Large	120	120	60	60
Small	120	120	60	60

(b) Departure (row) vs. Arrival (col)

	B75*	Heavy	Large	Small
B75*	60	60	60	60
Heavy	70	70	70	70
Large	60	60	60	60
Small	50	50	50	50

(c) Arrival (row) vs. Departure (col)

	B75*	Heavy	Large	Small
B75*	28	28	28	28
Heavy	24	24	24	24
Large	28	28	28	28
Small	40	40	40	40

arrival on runway 23 begins to exit its runway. Also, aircraft that wish to depart or arrive on runway 18C must wait until all runway crossing traffic is cleared as well, and vice-versa.

Any sequence of departure aircraft will be separated based on the immediate departure fix. This fix

separation takes two forms: (1) standard separation and (2) miles-in-trail (MIT). A standard departure fix separation between any two departure aircraft using the same fix is constant across all fixes. For example, aircraft which follow each other to the same fix must be always be separated by at least 80s.

In addition to standard fix separations, SOSS can simulate larger separations across departure fixes. This functionality is used to simulate MIT [12] restrictions[b]. For example, if departure fix JACAL is deemed to have an MIT restriction of 240s, consequently any two departure aircraft using JACAL must be separated by at least 240s.

D. Uncertainty Module

For SOSS to model realistic events, it uses an internal uncertainty module. This uncertainty module is composed of two components that correspond to two main types of uncertainty: (1) movement uncertainty and (2) time uncertainty.

1. Movement Uncertainty Error

Movement uncertainty error involves perturbations of aircraft speeds. Recall that SOSS has ramp, active movement area, and departure queues as its three main areas for aircraft to move. Among each of these areas, SOSS can perturb an aircraft's speed independently.

Using a mean speed error, standard deviation, and distribution function SOSS calculates the perturbations in aircraft speeds. Each main area on the airport surface has its uncertainty controlled by its own mean error, standard deviation, and distribution function. This allows a user to modify speed errors independently on various areas on an airport surface.

An additional feature of the movement uncertainty error is the ability to control the mode of operation. The mode of operation can apply uncertainty (1) every simulation time step, (2) every time an aircraft accelerates, or (3) at a prescribed call interval. If, for example, uncertainty error is applied every time an aircraft accelerates, then each time an aircraft decelerates or accelerates a new speed is *drawn* from the distribution.

2. Time Uncertainty Error

Time uncertainty error in SOSS provides the basis for creating pushback duration and flight readiness uncertainty.

Pushback duration is defined as the amount of time between the moment a departure aircraft pushes back and the moment it begins taxiing. Pushback duration is the time that passes as the aircraft waits until it can begin taxiing from an area near its gate to the runway. Usually, this corresponds to a delay that occurs because an aircraft's crew must adequately prepare the flight prior to actual taxi. Like movement uncertainty, pushback duration uncertainty is controlled by a distribution with a mean and standard deviation. However, in the case of pushback duration uncertainty the operational modes do not apply.

Flight readiness corresponds to the events that take place prior to an aircraft being able to pushback. While scheduled departure time indicates a desired pushback time, it's often the case that aircraft are late (and sometimes early) to pushback due to mechanical issues, passengers, crew support, pilots, etc.To make the model closer to reality, a perturbation on the time when an aircraft can pushback is allowed. SOSS, again, allows the user to select a distribution and its parameters to characterize flight readiness uncertainty.

E. Scheduler Interface

In addition to the core-functionality, SOSS provides a mechanism for schedulers to directly control airport surface traffic. To control traffic a scheduler must define a (1) time horizon, (2) call interval, and (3) domain. The time horizon is used to determine how far in the future a scheduler should include aircraft. For example, a time horizon of 10 minutes implies that a scheduler will only schedule those aircraft currently present on the airport surface and those predicted to be on the surface in 10 minutes. The call interval is used to define how often the scheduler is called. For instance, a call interval of 10 seconds means that every 10 seconds the scheduler is asked for a schedule. When the scheduler is asked for a solution SOSS will pause the simulation

[b]Usually MIT restrictions are given in units of miles of separation, but SOSS enforces separations using time.

and wait for a solution. Finally, the domain is the area of the airport that the scheduler is focused on. For instance, if 18C is specified then only aircraft using 18C will be passed to the scheduler from SOSS.

With these three parameters defined, a scheduler can request that SOSS release aircraft at nodes on the airport surface at particular times. For example, a scheduler could ask SOSS to release departure aircraft from gates at prescribed times.

IV. Experiment

A. Experiment Setup

1. Demands

This experiment is based on a two hour ASDE-X derived scenario from CLT. Using this scenario thousands of other scenarios are created by sampling earliest gate pushback times for departures and runway exit times for arrivals. Our sampling is accomplished using a uniform distribution from -100 to 100 seconds. For instance, the actual scenario may suggest that an arrival exits the runway at simulation time 100, but with a sampled error of 10s, the new scenario will have the arrival exit at 110s.

For all scenarios, CLT operates in south-flow configuration, where departures primarily depart on runways 18C and 18L, and arrivals land on runways 18R and 23. The primary test runway is runway 18C. The average (across scenarios) departure demand for runway 18C is shown in Fig. 5.

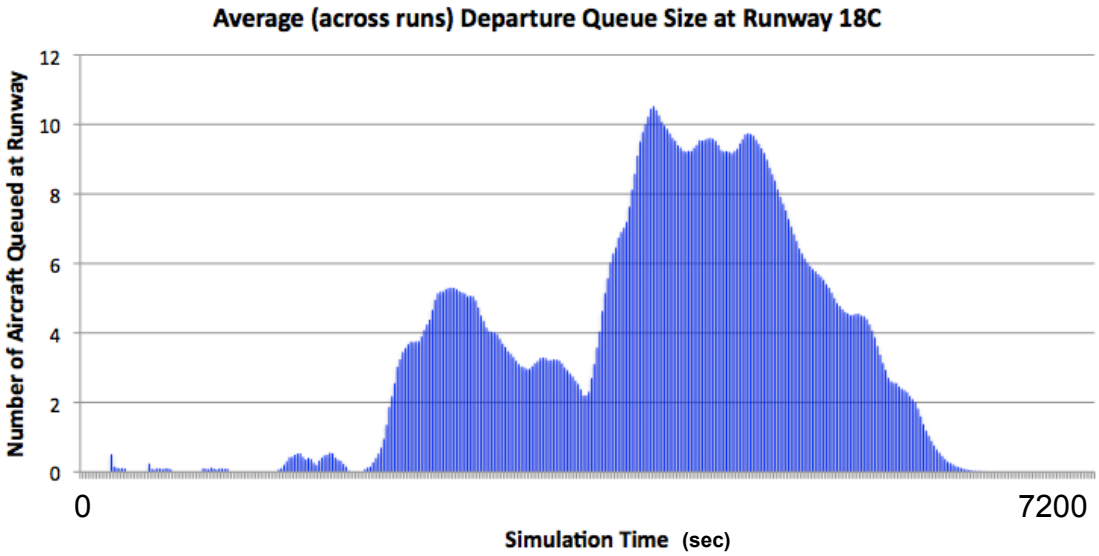

Figure 5: Average (across all scenarios) size of the runway queue at runway 18C.

Other important statistics (and constant across all runs) of the scenarios are summarized in Tables 2a and 2b. Table 2a shows the number of departure and arrival aircraft using each runway. Moreover, Table 2b indicates the counts across each departure fix for runway 18C.

2. SOSS Conflict Detection and Resolution Settings

SOSS is configured with taxi conflict detection and resolution *only* in the active movement area. Additionally, SOSS enforces all separations on the runways using the inter-aircraft separation matrices described in section III. As for taxi conflict detection and resolution, the static separation component (e.g., α) of equation (11) is set to 4 meters. β or the dynamic component is set to 1s.

3. SOSS Default Scheduler

While each scheduling algorithm makes the final decision on when an aircraft will leave its gate, SOSS takes over as soon as the aircraft releases from its gate. The logic SOSS uses is fairly simple by performing strictly first-come-first-served.

Table 2: Various scenario statistics.

(a) Counts of aircraft on runways at Charlotte Douglas International Airport for a given scenario.

	18C	18L	23	18R
# Departures	49	46	0	0
# Arrivals Landing	13	0	35	26
# Arrival Crossers	26	0	0	0

(b) Counts of aircraft across departure fixes for runway 18C

DEBIE	JACAL	ZAVER	SUG	SPA	MERIL
15	14	12	1	1	6

This means that once a scheduler decides to release an aircraft from its gate, any sequencing decisions at taxi intersections or runways is accomplished on a first-come-first-served basis. This assumption supports the assumption that ramp control can only release aircraft from gates and not make any active movement area scheduling or runway scheduling decisions, which are left to ground and local controllers. In reality, however, ground and local controllers tend to be more efficient than first-come-first-served at runways.

4. Scheduler Interface

SOSS calls each scheduler at a 10s call-interval. Each time SOSS calls the scheduler, it pauses the simulation until the scheduler provides a set of gate release times. Also each scheduler is provided all aircraft on the airport surface and all those aircraft predicted to be ready in 10 minutes. Note, this is not realistic since the flight readiness uncertainty would cause errors in this prediction. However, in this experiment all the schedulers do not face any flight readiness uncertainty.

B. Experiment Goals

The goals of this experiment are to assess the benefit of using three different surface scheduling techniques : an MILP based scheduler, a heuristic algorithm, and a baseline scheduler under variety of testing conditions. In order to accomplish this task, hundreds of simulations are performed on each scheduler, certain testing conditions are provided, and metrics are calculated

1. Testing Conditions

This experiment consisted of two independent testing conditions. The first condition is uncertainty, and the second condition is a miles-in-trail restriction.

Uncertainty: Uncertainty in these simulations is either on or off. When uncertainty is on, simulations have only time uncertainty in the pushback duration. Using the uncertainty data collected in [13], pushback duration is modeled using a uniform distribution with a 73 second standard deviation and a mean of zero. This means that the pushback duration can be any where from -126 to 126 seconds. However, the pushback duration uncertainty is always positive according to [13] and therefore samples from 0 to 126 seconds are used. The parameters for uncertainty are summarized in Table 3.

Table 3: Uncertainty parameters

	Distribution	Mean	Standard Deviation
Ramp (movement)	N/A	0	0
Taxi (movement)	N/A	0	0
Departure Queue (movement)	N/A	0	0
Pushback Duration (time)	Uniform	0sec	73sec

Miles-in-trail: Miles-in-trail separation in these simulations is either on or off. When MIT separations are on, a 240s MIT is applied across JACAL. Note that this constraint only applies to those aircraft which depart on 18C. According to Table 2b the percent of aircraft who use 18C and JACAL is about 28.6%.

2. Metrics

To determine the benefit of using a scheduling algorithm, two metrics are evaluated: (1) average taxi time per aircraft by region and (2) average delay per aircraft by region. These regions are defined as All, Gate, Ramp, Taxi, Queue, Departure, Arrival, and Crossing, and correspond to those links on the airport node-link model described in s=Section III.

The average taxi time per aircraft by region is reported as the total taxi time for that airport region divided by the total number of aircraft. To calculate the taxi time for any region for any aircraft, the total amount of time an aircraft spends in that region is reported. This logic, however, is slightly modified for the case in the ramp area. Since the gate node is also part of the ramp area but an aircraft is not actively taxiing while waiting at a gate, it is excluded from the taxi-time calculation. Given that aircraft have their engines turned off while at gates, this means taxi-time can be thought of as the amount of time an aircraft has its engines on.

The average delay per aircraft by region is reported as the total delay for that airport region divided by the total number of aircraft. The total delay for any region for any aircraft is calculated as the actual time an aircraft spends in that region minus the estimated least time duration the aircraft could spend in the region. The estimated least time duration is calculated using the nominal unimpeded taxi speed of the aircraft for that region.

C. Scheduler Calibration

In order to perform the experiments and collect the metrics, the MILP and heuristic need to be calibrated. These calibrations are performed using the condition with no uncertainty and MIT restriction active. The idea is to get an estimate of N_r and W_r for the most limiting conditions. The inclusion or exclusion of uncertainty while calibrating will have dramatic effects on the results, but those are discussed in Section V.

First, a calibration of N_{18C}, or the maximum number of departure aircraft allowed to be away from gates, is accomplished for the heuristic. Fig. 6 shows the average total delay per aircraft as a function of N_{18C}. Fig. 6 indicates that at about $N_{18C} = 9$, the average total delay per aircraft begins to level off. Also drawn on these figures is the first standard deviation error bars showing the spread of the data according to successive runs. It should be clear that with $N_{18C} = 9$, one would expect a reasonable amount of total delay. Hence, for all other tests $N_{18C} = 9$.

Next, a calibration of W_{18C}, or the desired wait time in the departure queue for runway 18C, is accomplished for the MILP. Fig. 7 shows the average total delay per aircraft by region as a function of W_{18C}. Fig. 7 provides insightful results regarding the calibration of the MILP-based scheduler's W_{18C} parameter. While these scenarios have no explicit uncertainty, there is still an implicit uncertainty caused by the fact that the scheduler cannot control any traffic that is actively taxiing. This effect is noticeable since a higher queue duration of 200s exhibits the lowest total delay (e.g., "All" column).

To explain why a 200s queue duration is needed, consider the following argument. Since no scheduler has direct control over any of the activities that occur post-pushback, the MILP scheduler cannot control when any aircraft actually uses the runway. So while the MILP scheduler may think an aircraft will queue up at the departure runway, it may end up leaving on a first-come-first-served basis. This results in excessive delays for other aircraft waiting at their gates, since those aircraft should have queued at the runway behind the departure to prevent the runway from going dry. To follow this analysis, look at the total delay (or the All column) and compare that with the delay taken at the gate. It appears that the excessive delays with W_{18C} set to 60 and 120 seconds is caused by the extra delay spent at the gate. This means that the MILP is holding departure aircraft for too long at the gate, assuming there is enough buffer at the runway to justify the holding. To mitigate against these effects and other uncertainty provided in later scenarios, W_{18C} is set to 200s.

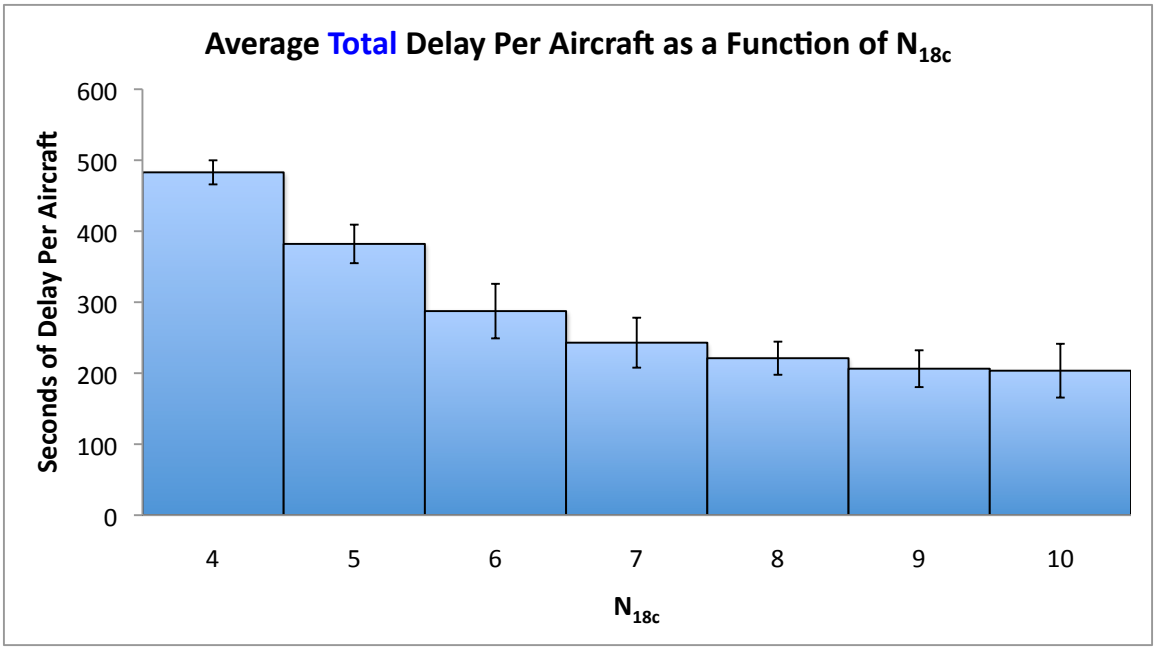

Figure 6: The average total delay per aircraft as function of N_{18C} for the simple gate holding heuristic.

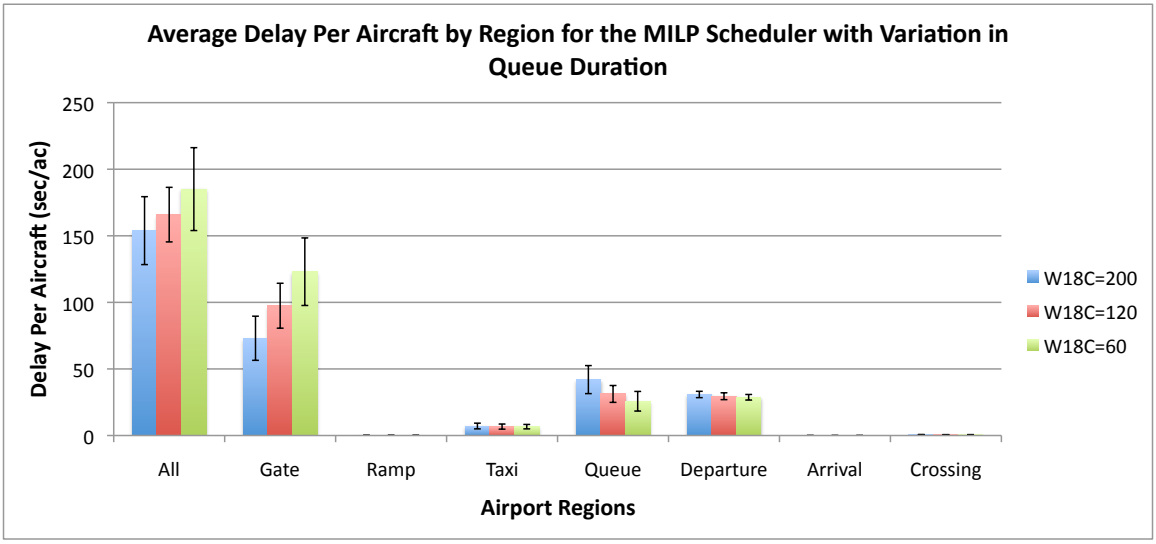

Figure 7: The average total delay per aircraft as function of W_{18C} for the MILP-based scheduler across various regions.

V. Results

With the schedulers calibrated and the experiment described, this section describes results for various testing conditions. Results are decomposed into four subsections. Section V.A provides results for the condition with no uncertainty and no MIT restrictions. Section V.B provides results for the condition with no uncertainty and MIT restrictions. Section V.C provides results for the condition with uncertainty and no MIT restrictions. Finally, section V.D provides results for the condition with uncertainty and MIT restriction active.

A. No Pushback Duration Uncertainty and No MIT Restriction

First, consider the average delay per aircraft by region in Fig. 8. With no pushback duration uncertainty and no MIT restriction, all schedulers achieve similar total delay distributions. However, the places where delays are attributed are different for the each algorithm. For the MILP based scheduler, the amount of delay spent at the gate is highest, while there are less delays in the departure queue than any other solution. For the heuristic algorithm, delays are also taken at the gate, but are less than the MILP scheduler. There is an average of 20 seconds less time spent at the gate with the heuristic algorithm than the MILP scheduler. The baseline algorithm achieves both the highest average delay per aircraft for any region, except for the gate. This is due to the fact that the baseline simply releases the aircraft as soon as possible.

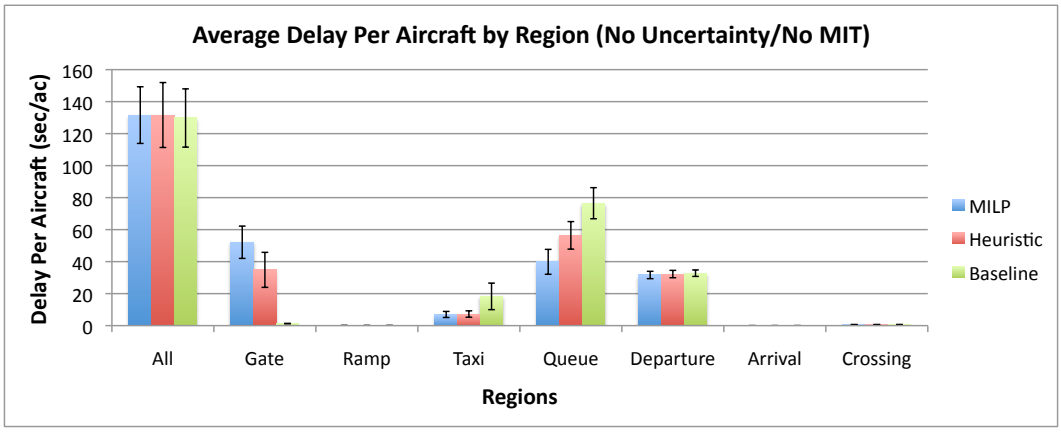

Figure 8: The average delay per aircraft by region for the MILP, heuristic, and baseline algorithms for hundreds of scenarios, where no pushback duration uncertainty is present and no MIT restriction is active. This figure also shows the first standard deviation.

Next, consider the average taxi time per aircraft by region in Fig. 9. Since there is no ramp conflict detection and resolution, the ramp taxi speeds are always the same (e.g., no aircraft is delayed in the ramp). However, notice that the taxi and queue areas show variations in average taxi time across the different schedulers. The MILP and heuristic have similar average taxi time on the taxiways, and the baseline is slightly higher than both the MILP and heuristic. In the departure queue, the MILP has lower taxi times than both the heuristic and baseline.

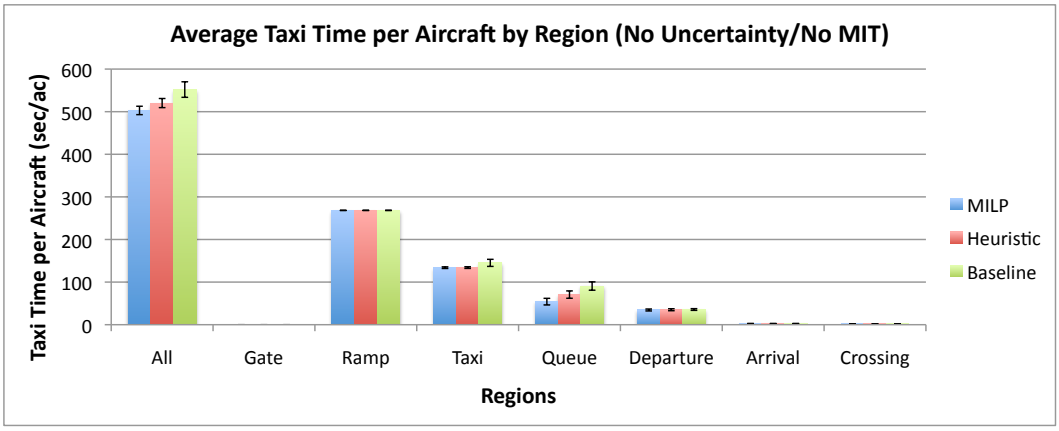

Figure 9: The average taxi time per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where no pushback duration uncertainty is present and no MIT restriction is active. This figure also shows the first standard deviation.

Figures 8 and 9 suggest that the MILP is able to keep a smaller queue than both schedulers and a lower amount of taxi time. Given that there is no MIT restriction for these results and the total delays are equivalent for all schedulers, these findings are a direct result of the buffering strategy used in the MILP

(e.g., W_{18C}) as compared to the baseline (non-existent) and heuristic (e.g., N_{18C}).

B. No Pushback Duration Uncertainty and With MIT Restriction Active

Fig. 10 shows average delay per aircraft by region for the case with no pushback uncertainty and an MIT restriction. For these test conditions, the MILP results in less total delay than any other scheduler. Additionally, the MILP achieves the lowest average delay spent away from the gate. The MILP showing less average delay than any other scheduler for these test conditions should indicate that this is a result of adding the MIT restriction to the scenarios. With consideration of Fig. 8, where there is no MIT restriction, the MILP performed the same as the other schedulers. Since an MIT restriction provides opportunities for sequencing to reduce total delay, differences between the MILP and other schedulers is expected. For example, looking carefully at the departure node delay (e.g., the amount of delay spent at the last node of a departure's route), one can see that the MILP actually reduces this time. These reductions translate to reductions in time of aircraft spent in the departure queue.

Although Fig. 10 indicates the MILP produces the lowest average delay, the heuristic also produces lower delays than the baseline. The amount of taxi delay produced by the heuristic is identical to the MILP, which is approximately 5 sec per aircraft, 12.5% of the taxi delay produced by the baseline.

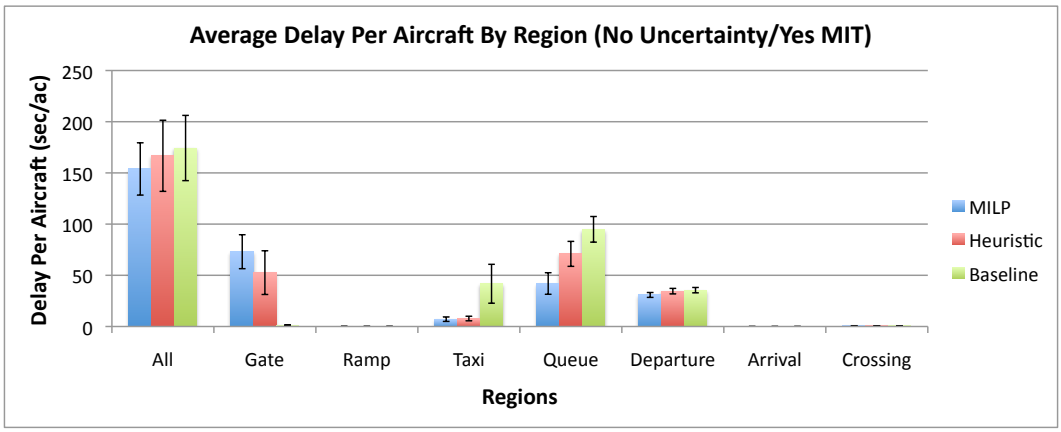

Figure 10: The average delay per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where no pushback duration uncertainty is present and an MIT restriction is present. This figure also shows the first standard deviation.

Fig. 11 indicates the average taxi time per aircraft is least for the MILP scheduler. Specifically, the average taxi time per aircraft in the departure queue is smallest for the MILP. Moreover, the average taxi time on the taxiway for the heuristic and MILP is smaller than the baseline.

The small standard deviations in taxi times is due to the fact that all aircraft, across various scenarios, have identical routes. The strong variations occur when there is traffic or congestion. For instance, the baseline shows relatively larger variations in taxi time on the taxi links than any other scheduler. This is because the baseline algorithm creates long queues at the runways, which leaves aircraft congested on the taxi links. Again, the departure queue shows variations for all schedulers due to congestion near or at the runway.

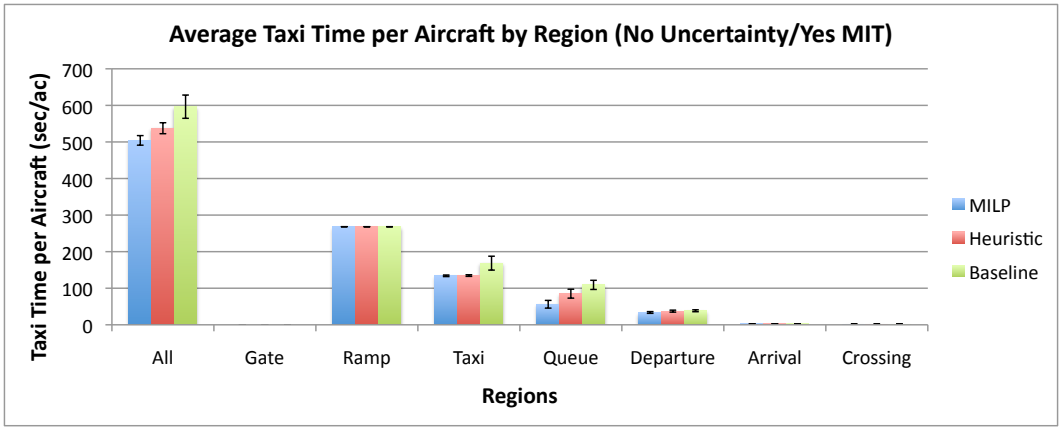

Figure 11: The average taxi time per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where no pushback duration uncertainty is present and an MIT restriction is present. This figure also shows the first standard deviation.

C. With Pushback Duration Uncertainty and No MIT Restriction

Figures 12 and 13 show the average delays and taxi times by region, respectively, for the case with uncertainty in pushback duration and no MIT restriction.

The average delay for these scenarios is completely reversed from Section V.B, where the average delay is highest for the MILP scheduler and lowest for the baseline. This result is likely an artifact of the buffer calibration used on the MILP and heuristic scheduler. Both schedulers were calibrated using scenarios where there was no uncertainty and an MIT present, which is completely orthogonal to the conditions that produced the results in Fig. 12.

Carefully examining Fig. 12 for the MILP, one can see that most delay is concentrated at the gate. This suggests that the MILP scheduler is holding aircraft back at the gates excessively. Again, this confirms the idea that the buffers used in the MILP are poorly chosen for these scenarios. Also, it suggests that the buffer is too low for these scenarios, since delays are mostly at the gates.

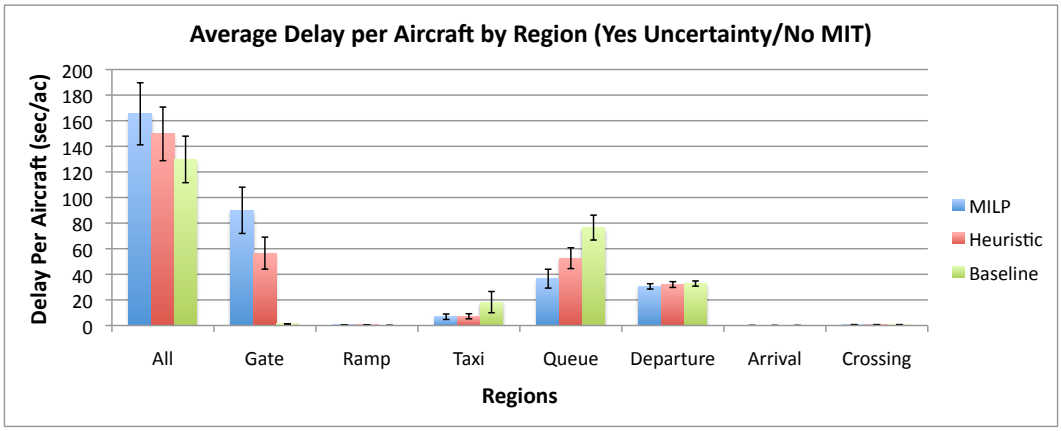

Figure 12: The average delay per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where pushback duration uncertainty is present and no MIT restriction is present. This figure also shows the first standard deviation.

With the average delays higher, the MILP shows the least average taxi time in Fig. 13. Comparing Fig. 13 with 9, one can see that uncertainty has no affect on the trends of average taxi time across all schedulers. The same relative benefits in taxi time are identical in both figures.

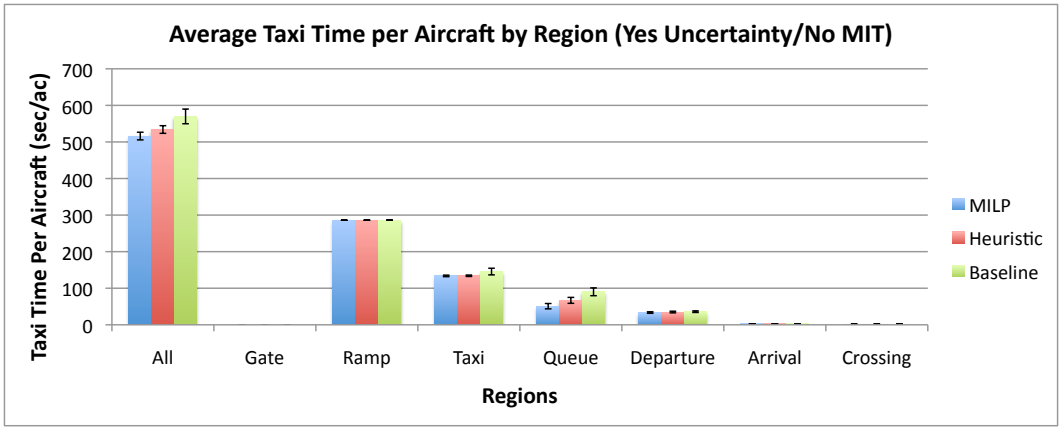

Figure 13: The average taxi time per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where pushback duration uncertainty is present and no MIT restriction is present. This figure also shows the first standard deviation.

D. With Pushback Duration Uncertainty and MIT Restriction

Finally, this section provides the result for the case where pushback uncertainty and an MIT restriction are present. Figures 14 and 15 show the average delay and taxi time, respectively.

Introducing the MIT restriction for the case with pushback uncertainty results in the MILP producing the least average delay. This result is completely different than those produced under the conditions in Section V.C. This result substantiates that simulating conditions closer to those used in calibrating a scheduler produces higher benefits.

There is substantial variation in average delay to which one could argue there is no evidence that the MILP reduces delays. However, looking at the delays across all regions, it is clear that the MILP strongly produces the least delays for the departure queue than any other scheduler. The large variations caused in the average delay across all regions (e.g., "All") could be caused by the amount of gate hold time induced by the MILP scheduler. Consequently, using the MILP would reduce, on average, the amount of time an aircraft would have its engine on.

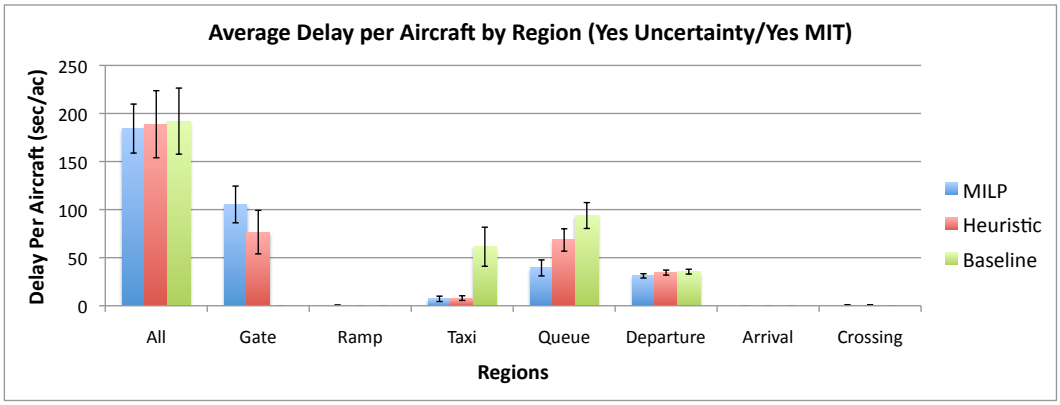

Figure 14: The average delay per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where pushback duration uncertainty is present and an MIT restriction is present. This figure also shows the first standard deviation.

Fig. 15 shows the average taxi time per aircraft by region for the condition where pushback duration uncertainty and an MIT restriction are present. This figure indicates that the MILP produces the least average taxi times. Also, the figure suggests that the MILP produces lower taxi delays in the departure queue than any other scheduler.

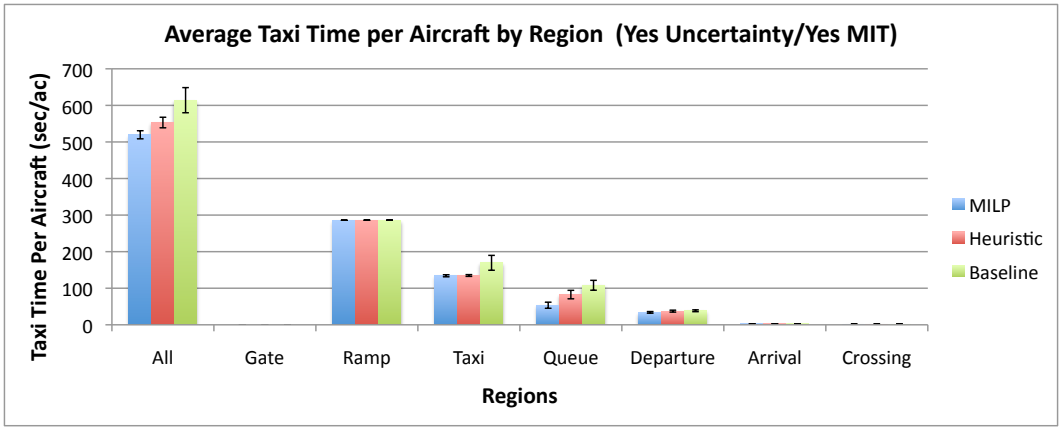

Figure 15: The average taxi time per aircraft by region for the MILP, heuristic, and baseline scheduler for hundreds of scenarios, where pushback duration uncertainty is present and an MIT restriction is present. This figure also shows the first standard deviation.

VI. Conclusion

The effort described in this paper developed an ensemble of scenarios to test three algorithmic concepts for metering aircraft at gates using NASA's fast-time simulation tool Surface Operations Simulator and Scheduler. Using Charlotte Douglas International Airport as a test-bed for conducting simulations, the authors study four test conditions: (1) without pushback duration uncertainty and without MIT restriction, (2) without pushback duration uncertainty and with MIT restriction, (3) with pushback duration uncertainty and without MIT restriction, and (4) with pushback duration uncertainty and with MIT restriction.

The condition without pushback duration uncertainty and without MIT restriction resulted in similar average delay per aircraft across the MILP and heuristic algorithms, but with the MILP producing less taxi-times. The baseline performs the worst for these conditions, with higher delays and taxi-times. The condition without pushback duration uncertainty and with MIT restriction resulted in the MILP producing the lowest average delay and taxi time per aircraft. Moreover, the heuristic performs better than the baseline for these conditions. The condition with pushback duration uncertainty and without MIT restriction resulted in the baseline algorithm producing the lowest average delay. The heuristic and the MILP produced higher delays, but lower taxi times than the baseline. This suggests that the scenarios used to calibrate W_{18C} and N_{18C} play an important role in the performance of the MILP and heuristic schedulers, respectively. The condition with pushback duration uncertainty and with an MIT restriction resulted in the MILP producing the lowest taxi times with equivalent average delays to any other algorithm. However, close inspection of the delays showed that the MILP produced the lowest departure queue delays, which translated to the lowest amount of time an engine would need to be on. This result is, in fact, similar (if not equivalent) to the MILP producing the lowest taxi times.

In summary, using the MILP and heuristic schedulers are only as good as their calibration for creating a buffer of aircraft at the runway. For example, the MILP scheduler is robust against uncertainty when the queue duration (W_{18C}) buffer is chosen carefully. As a future study, it would be insightful to study various buffer concepts.

Lastly, runway sequencing appears to be efficient only when separation between successive aircraft at runways is large. For instance, when there is no MIT restriction, the MILP performs no better than any other scheduler for average delay; though theoretically, there should be advantages to sequencing with the standard separation matrices. Most of the advantages are overshadowed by the fact that the majority of aircraft that operate at CLT are large. As another study, the amount of separation and uncertainty required for efficient sequencing should be investigated under various airports and configurations.

References

[1] Simaiakis, I., "Analysis and Control of Airport Departure Processes to Mitigate Congestion Impacts," TRB Annual Meeting 2010.

[2] Airport Surface Detection Equipment Model X, Saab-Sensis Corporation. http://www.saabsensis.com/products/airport-surface-detection-equipment-model-x-asde-x/

[3] Bhadra D., Knorr D., and Levy B., "Benefits of Virtual Queuing at Congested Airports Using ASDE-X: A Case Study of JFK Airport," 9th USA/Europe Air Traffic Management Research and Development Seminar, July 2011.

[4] Simaiakis I., "Modeling and control of airport departure processes for emissions reduction," Massachusetts Institute of Technology Thesis, August 2009.

[5] Jung, Y., et al., "Performance Evaluation of a Surface Traffic Management Tool for Dallas/Fort Worth International Airport," Air Traffic Management R&D Seminar, June 14-17 2011, Berlin, Germany.

[6] Windhorst, R., et al., "Towards a Fast-time Simulation Analysis of Benefits of the Spot and Runway Departure Advisor," 14^{th} Aviation Technology, Integration, and Operation (ATIO) Conference, September 2012, Indianapolis, Indiana.

[7] Griffin, K., Aditya, S., et al., "Benefits Assessment of a Surface Traffic Management Concept at a Capacity-Constrained Airport," 14^{th} Aviation Technology, Integration, and Operation (ATIO) Conference, September 2012, Indianapolis, Indiana.

[8] 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.

[9] Montoya, J., Rathinam, S., Wood, Z., "Multi-Objective, Departure Runway SCheduling Using Generalized Dynamic Programming," IEEE Intelligent Transportation Systems, In review beginning July 2012.

[10] Malik W. A., Gupta G., and Jung Y., "Managing departure aircraft release for efficient airport surface operations," AIAA Guidance, Navigation, and Control Conference, Toronto, Canada, August 2-5, 2010.

[11] Griffin, K., et al. "Final Report: Adaptation of a Surface Traffic Management Tool to Multiple, Capacity-Constrained Airports," Document No.:20115743-D10, Version: 1, February 15th 2013, Contract: NNA11AC50C, Prepared for Nasa Ames Research Center

[12] 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.

[13] Capps, A., Day, K., et al., "Impact of Departure Prediction Uncertainty on Tactical Departure Scheduling System Performance," 14^{th} Aviation Technology, Integration, and Operation (ATIO) Conference, September 2012, Indianapolis, Indiana.