Novel, Multidisciplinary Global Optimization under Uncertainty

Phase II Final Report

Version: 1

October 31, 2016

Cooperative Agreement: NNX15AR29A

Prepared for:

NASA/Shared Services Center (NSSC)

Building 1111, Jerry Hlass Road

Stennis Space Center MS 39529-0001

Prepared by:

ARCHITECTURE TECHNOLOGY CORPORATION

SPECIALISTS IN COMPUTER ARCHITECTURE

Architecture Technology Corporation

P.O. Box 24859

Minneapolis, Minnesota 55424

REVISION HISTORY

Version	Date	Author(s)	Change Sections	Description
1	October 31, 2016	Greg Carr, Matthew Stillerman, Mark Peters, Valentino Felipe, Sebastian Timar – Architecture Technology Aniruddha Basak, Priya Sundararajan, Erika Menezes, Vinodh Paramesh, Ole Mengshoel – Carnegie Mellon University Nikunj Oza, Bryan Matthews, Ilya Avrekh – NASA	All	Initial Release

Table of Contents

1 Abstract

Novel, Multidisciplinary Global Optimization Under Uncertainty is a two year research effort sponsored by the NASA Leading Edge Aeronautics Research for NASA (LEARN) project. The research focused on developing and evaluating the Probabilistic Robust Optimization of Complex Aeronautics Systems Technology (PROCAST), which combines methods from statistical modeling – Bayesian Networks (BNs) for probabilistic modeling and prediction – and complex adaptive systems – a method of constrained optimization. The goal of PROCAST is to generate solutions for complex system problems characterized by three key attributes: multiple competing objectives, uncertainty impacts, and network effects. A problem that displays these three characteristics is the management of air traffic in busy, highly interconnected regions of the National Airspace System (NAS) such as metroplexes. A metroplex consists of multiple busy airports in close vicinity of each other. Metroplex airports and airspace are key capacity bottlenecks within the NAS. In this application, PROCAST aims to provide optimized and de-conflicted sequencing and scheduling of arriving/departing flights that is robust to uncertainties such as forecasting errors, and variations in air traffic operations.

Over the course of the two year research effort we developed a system architecture and simulation framework to evaluate the viability of PROCAST applied to integrated arrival, departure, and surface traffic scheduling in the New York metroplex – comprised of three busy airports: John F. Kennedy International (JFK), Newark Liberty International (EWR) and LaGuardia Airport (LGA). Accomplishments include the development of airport surface and terminal models for NASA's Surface Operations Simulator and Scheduler (SOSS), the development of an integrated, multi-airport arrival and departure scheduler, the development of BNs to probabilistically model taxi-times at the airports, and a method to account for probabilistic transit times into traffic scheduling. Improving the modeling and predictability of taxi times and wheels-off times, and traffic scheduling methods which are robust to their uncertainties, are key factors in improving the efficiency of airport operations. We evaluated the effectiveness of using BNs in arrival-departure scheduling compared with other, simpler probability models, and with no probability modeling as well. The use of probabilistic modeling for taxi-time prediction makes the integrated arrival and departure scheduling more robust to uncertainties and therefore more likely to be useful in future air traffic management (ATM) decision support tools.

The results of this research demonstrate that the PROCAST scheduling using probabilistic BN models of taxi time for departure trajectory prediction was able to significantly improve the performance of departure traffic from JFK, EWR and LGA airports for the two different real-world traffic days evaluated, in comparison to scheduling using deterministic models, or simple probabilistic Gaussian models, of departure taxi time. The PROCAST scheduling with BN models of taxi time was able to significantly reduce the average total delay of departures at each airport, and likely improves the departure throughput at each of the airports, compared to the deterministic modeling. Average total delay reductions ranged from 4.0 minutes for LGA departures for a 2-hour period of traffic on 3/16/2012 to 29.0 minutes for JFK airport for a 2-hour period of traffic on 7/25/2012. While the results are promising, our hypotheses of improved traffic performance through incorporation of uncertainty into traffic planning, increased accuracy of transit time uncertainty modeling, and increased sampling uncertainty models were not conclusively confirmed, and further investigation is warranted. In addition, there are many areas for future research, including alternative approaches to BN modeling of aircraft transit, holistic integration of 4D trajectory prediction for scheduling and accounting for the influence of scheduling on the predicted trajectory, implementation of alternative methods for selecting the "statistically best" scheduling solution among the alternative set of scheduled traffic futures, or otherwise incorporating the probability information into the traffic scheduling, and enhancing SOSS to rigorously model multi-airport interactions and terminal airspace traffic.

2 Project Overview

In Phase I we performed a PROCAST proof-of-concept experiment applied to the management of traffic at a single airport in the New York metroplex. The Phase I effort included modeling, simulating, and evaluating the performance of PROCAST on the JFK airport surface and terminal airspace. We selected JFK because of the operational challenges it faces such as significant arrival and departure delays, and complex airspace and surface arrival-departure interactions. The proof-of-concept experiment compared the performance of PROCAST operations against baseline operations that represent traffic under current-day traffic management procedures. We simulated current operations for JFK, and simulated future concepts and procedures using PROCAST and SOSS. We then compared the metrics from the two simulations to quantify the benefits provided by PROCAST. The motivation for the Phase I research, as well as a detailed description of the assumptions, methods and results of the Phase I proof-of-concept experiments are documented in the Phase I Final Report [SS15] and two conference papers [AS14] [AS15]. Our Phase I proof-of-concept experiments showed that PROCAST can provide significant benefits at a single-airport, and the potential benefits could be much larger if PROCAST is applied to coordinate traffic at multiple metroplex airports.

The primary objective of Phase II is to expand the application of PROCAST to scheduling traffic to multiple airports in the New York metroplex. Phase II includes enhancing SOSS by creating airport surface and terminal models of EWR and LGA [RW12]; enhancing PROCAST to perform integrated, multi-airport arrival and departure scheduling; enhancing the probabilistic Bayesian Network taxi-time models for JFK from Phase I; and creating Bayesian Networks to probabilistically model taxi-times at EWR and JFK. Outreach to subject matter experts helps identify key factors to consider in taxi-time modeling and airport traffic scheduling and management. Improving the modeling and predictability of taxi times and wheels-off times is a key factor in improving the efficiency of airport operations. NASA is actively pursuing research into taxi-time prediction modeling, simulation, and evaluation. Accurate taxi time estimates support departure management strategies such as holding departures at their gates, optimizing the use of departure runways at origin airports, and improving arrival time prediction at destination airports [HL15]. It should be noted that in Phase II we no longer use an explicit optimization routine in the PROCAST solution architecture because the benefit did not justify the computational complexity. Instead, we focus on developing and investigating the use of BNs to probabilistically model and predict taxi-times on the airport surface within the PROCAST framework. We evaluate the effectiveness of using BNs in arrival/departure scheduling compared with other, simpler probability models, and with deterministic transit time modeling. As in Phase I the BNs provide a means of incorporating probabilistic modeling into arrival and departure scheduling in a computationally efficient manner. Our Phase II work was organized into six high-level tasks depicted in Figure 1.

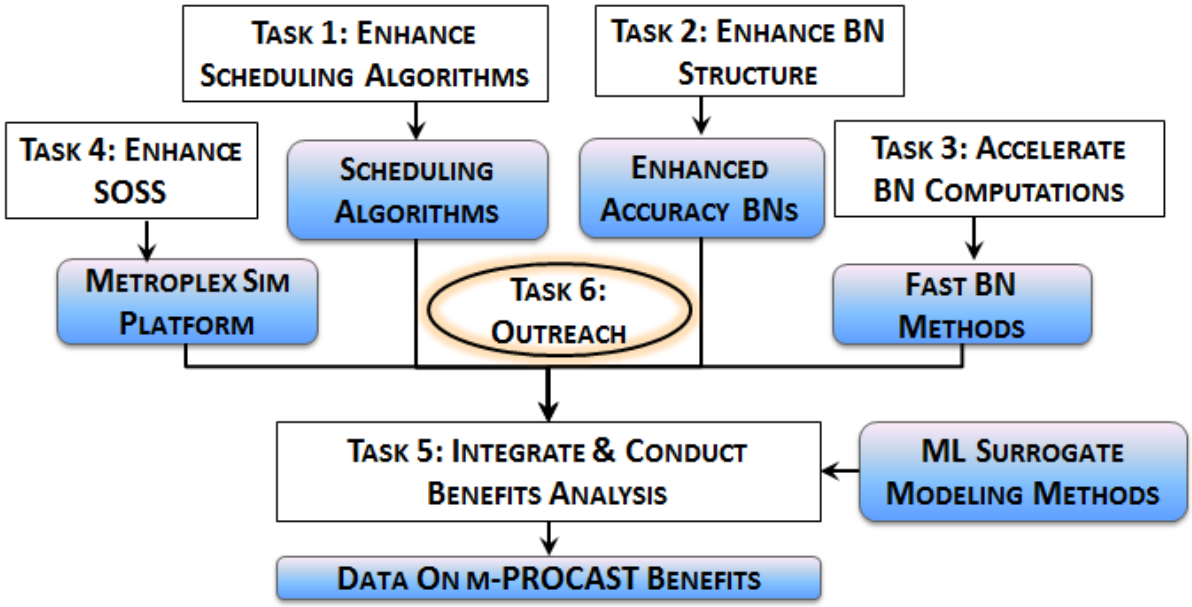


Figure 1. Phase II Technical Tasks

3 PROCAST Architecture and Bayesian Networks

One of the main goals of our Phase I and Phase II research efforts is to make the management of surface-terminal air traffic operations robust to uncertainties such as unpredictable errors in the gate pushback times, taxi times, and terminal airspace traversal times forecasted for flights. To achieve this goal, we use BNs to incorporate uncertainty models into taxi-time modeling and prediction in a computationally efficient manner. A large number of random variables are required to sufficiently model uncertain network effects in the surface-terminal operations. Due to the complex nature of subsystem interactions, most of these random variables are dependent; so modeling their joint effects (i.e., joint probability distribution) requires a large number of Conditional Probability Distributions (CPDs). To address these challenges, we make use of BNs for incorporating probabilistic modeling into trajectory prediction and aircraft scheduling [AS15].

BNs model probabilistic relations among random variables. BNs model probabilities based on analysis of data. BN probability models are learned from training data, and then once trained, are used to make predictions about new data. BNs can be used for predictive modeling, pattern recognition, classification and regression [JH13]. BNs are probabilistic graphical models that represent a set of random variables and their conditional dependencies via a directed acyclic graph. BNs provide advantages over traditional probabilistic approaches by compactly encoding complex conditional distributions over a high-dimensional space and by providing a separation between solution method and model. In the context of the traffic management application, we use BNs to generate future trajectory forecasts based on the current measured positions and speeds of aircraft as well as reflected in simulated operational data [SS15].

The PROCAST architecture for the metroplex scheduling problem is shown in Figure 2. PROCAST first predicts a large number (e.g., thousands) of possible future traffic scenarios by sampling the BN models of transit time developed from analysis of historical traffic data. Each potential future traffic scenario consists of a possible future trajectory for each of the flights that is currently active within the airport surface or terminal airspace, or is expected to enter the airport surface or terminal airspace within a certain look-ahead time horizon. To predict the 4D trajectory of a flight, PROCAST assumes that the physical 3D route of the flight will remain fixed. PROCAST estimates the flight's crossing times at the

key nodes on its fixed 3D route by sampling the BN transit time models. The *predicted* 4D trajectories of a flight vary among the predicted future traffic scenarios in their crossing times at the key nodes of the flight's 3D route. The time-variations between the trajectories of a flight among the future traffic scenarios are the result of sampling the probabilistic BN models of transit time derived from historical traffic data. For this project, historical traffic data were obtained from previous SOSS simulations of multi-airport traffic; for operational implementation, traffic data may be obtained from recorded traffic surveillance data or other sources.

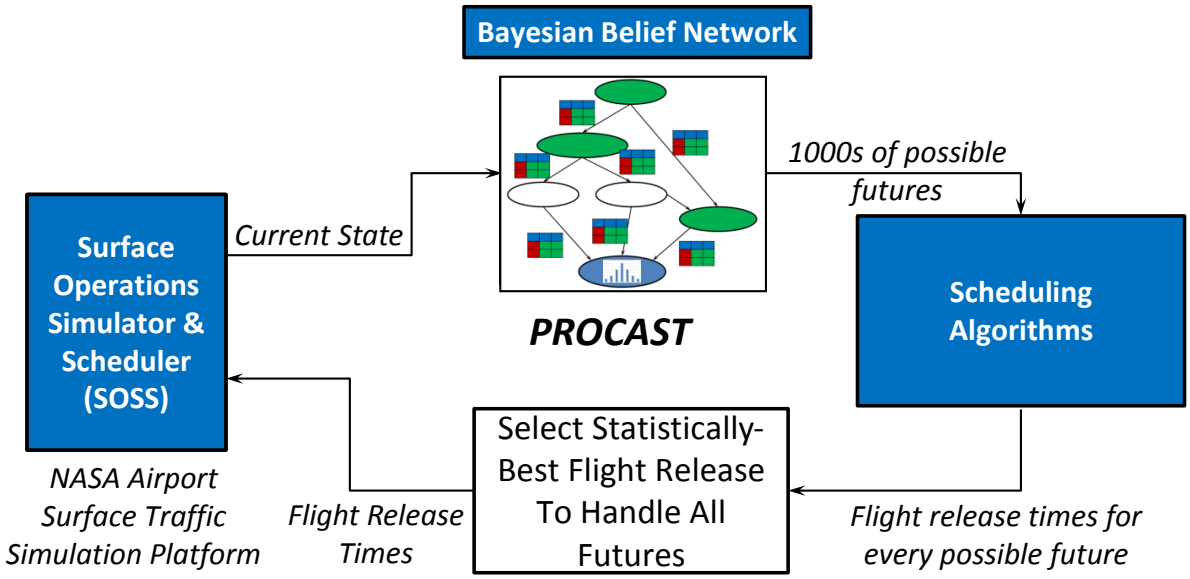


Figure 2. PROCAST Metroplex Integrated Arrival, Departure, Surface Scheduling Architecture

The randomness in the time dimension of the predicted 4D trajectories of a flight is the result of sampling the pre-computed probability distributions. The distributions describe the probability of a flight arriving to or leaving from certain fixed 3D nodes at a specific time. The probability distributions are computed using a BN that is trained from simulated data and conditioned on several influencing factors. Factors include the current measured traffic state of the airport-terminal system (i.e., the locations of all aircraft). The key points in the 4D trajectories of flights at which there is temporal uncertainty in the transit or occupancy time are the gates, spots, runway departure nodes, airspace interaction fixes, and departure/arrival fixes.

After generating a number of potential futures, PROCAST performs scheduling for each of the potential futures. That is, for each future, PROCAST revises the time dimension of the predicted 4D trajectory of each flight in the future as needed to comply with capacity limits of airport and airspace resources such as the runways and arrival and departure fixes. This produces a *planned* 4D trajectory for each flight. The basic objective of a scheduling algorithm in air traffic control automation is to match traffic demand and airport capacity while minimizing delays. The scheduling algorithm automatically determines (a) the order or sequence in which each aircraft should land, depart, or cross a particular fix, and (b) the time that each aircraft in the sequence should land, depart, or pass over a specified fix. The underlying scheduling algorithms use a First-Come First-Served (FCFS) approach. An earlier study [CA98] showed that scheduling aircraft according to an FCFS sequence based on estimated times of arrival produces a schedule that is considered to be both fair to air carriers and efficient in terms of minimizing delays that must be absorbed when demand exceeds capacity.

Once scheduling has been performed for each of the futures, probabilistic analysis is used to choose the "statistically-best" traffic schedule. PROCAST currently does this by selecting the "statistically best" future from among the set of scheduled futures. By applying this methodology, PROCAST selects a

schedule for the flights in the planning time horizon. The selected traffic schedule specifies a *planned* 4D trajectory for each flight which prescribes gate pushback and takeoff times for each departure or arrival fix crossing and landing times for each arrival. PROCAST implements this set of control actions, i.e., the collection of landing and fix crossing times for arrivals and takeoff and gate pushback times for departures (also called *flight release times*), to manage the traffic. In the experiments described in Section 7, the flight release times are sent to SOSS at the end of each scheduling cycle. The process is repeated at the next cycle using the latest available information.

3.1 Sources of Taxi Time Uncertainty at New York Airports

The purpose of PROCAST is to make the management of surface-terminal air traffic operations robust to uncertainties in the gate pushback times, taxi times, and terminal airspace traversal times forecasted for flights. To better understand real-world factors that influence uncertainties in the taxi times at JFK, EWR, and LGA, we held discussions with Ralph Tamburro of the Port Authority of New York/New Jersey (PANYNJ) and Bill Cotton of Cotton Aviation Enterprises (and former Manager of Air Traffic and Flight Systems, United Airlines).

In general, taxi times are affected by factors such as controller workload, fix restrictions and closures, runway queue length, one way alleyways in the ramp areas, and controller best practices to effectively manage departures such as sending consecutive departures to different fixes. For airplanes taxiing out at all three New York airports there are two factors influencing the time it takes to go from the gate to the runway liftoff: 1) the distance from gate to runway along with the taxi speed profile, and 2) the points on the taxi route that require clearance to proceed.

The taxi distance is measureable but the taxi speed varies by aircraft type and individual pilot preference. The pushback, engine start and initial movement processes take longer on very large aircraft. Four engines take twice as long to start as two, and heavy aircraft will accelerate and decelerate slower than light aircraft. Nominal taxi speeds range from 10 to 20 knots but may be slower or faster depending on taxiway surface condition, visibility, obstructions and other traffic, or activities in the cockpit.

At the three New York airports, continuous movement from the gate to takeoff is not possible because clearances are required for gate pushback, ramp departure, hold points in the taxi clearance, and runway takeoff. Each is described below in greater detail.

3.1.1 Gate Pushback

Once the airplane is buttoned up and ready to push, the pilot will call either ramp control or ground control to request pushback clearance and when received, and relay that to the tug crew to commence the pushback. While nominally that call would happen at the scheduled departure time, it is quite variable, from up to 10 minutes before departure time to perhaps hours late during a maintenance or weather irregularity. The distribution is skewed to the late side and the median and standard deviation should be available from the Department of Transportation's on-time departure statistics. Each airline also keeps track of the difference between scheduled and actual departure time and has at least a dozen codes indicating what management function the delay will be assigned to (both airline and ATC).

3.1.2 Ramp Departure

Clearance must once again be requested at the spot for transition from the non-movement area (ramp control) to the movement area of taxiways and runways (ground control). The granting of the taxi clearance from this point is influenced by the workload of the ground controller (being able to break into the frequency to request taxi) and by the actual presence of traffic on the taxiway that has right of way over the new request. Not all gates on the airport are used equally so the use of these spot transitions also varies with gate use. The ability of a controller to clear an aircraft from such a spot to the runway is very dependent on the spot's location with respect to other taxiway traffic.

3.1.3 Hold Points in the Taxi Clearance

These may be particularly busy taxiway nodes or nodes where the taxi route crosses another runway. They are indicated in the clearance as, "hold short of Bravo", or "hold short of Runway 4 Left". The amount of time spent waiting at these hold points for further clearance depends entirely on the volume of traffic on the taxiways, and runways to be crossed.

3.1.4 Runway Takeoff Clearance

The physical limit to the capacity of the runway is the requirement for a previous departure to be 6,000 feet into its takeoff roll (for a “Large” wake category aircraft) before clearing the next takeoff. This location approximates most airliner ground rolls before rotation so the practice is known as "nosewheeling". Unfortunately, the New York controllers rarely get to use this procedure because of departure constraints imposed on the tower by departure control or Air Route Traffic Control Center (ARTCC) requiring a greater separation between departures. Constraints requiring greater separation between departures are:

- Wake turbulence criteria
- Noise abatement
- Arrival crossing runway
- Departure fix restriction
- Non-standard weather departure route
- Overhead flow constraint
- Ground delay program for destination weather or sector load
- Ground stop due to weather

Determining the appropriate departure delays is challenging for controllers and traffic managers. The wake categories of the flights at these three airports must be learned as a function of the schedule and it is still hard to predict the takeoff order based on when they leave the gate. Some departure restrictions are static and can be gleaned from the FAA Air Traffic Control System Command Center website but the dynamic ones are only known at the time they are being used. Weather creates its own set of problems predicting the time, severity, geographic location and extent and the resulting re-routes and ground delay programs are as varied as the weather itself. Using a single “typical” bad weather day to juxtapose a good day leaves something to be desired. The extent of the delays imposed on takeoff clearance determines the amount of the queue developing at the runway as a function of demand during the day, and the resulting time to takeoff. These challenges highlight the need for the development of air traffic management decision support tools to help air traffic managers and air traffic controllers more effectively manage departure traffic at metroplex airports such as those in the New York area.

4 Probabilistic Multi-Airport Traffic Scheduling

This section presents the current implementation of the probabilistic multi-airport traffic scheduling in PROCAST. Probabilistic multi-airport traffic scheduling is achieved by 1) applying a FCFS scheduling methodology to a single predicted future for the set of aircraft being scheduled to generate a scheduled future for the set of aircraft, 2) applying this methodology to each of the predicted futures for the set of flights being scheduled, and 3) selecting a single scheduled future from among the set of scheduled futures as the basis for assigning gate pushback times to departures and arrival fix crossing times to arrivals. The gate pushback times of departures or fix crossing times of arrivals robustly comply with the inter-aircraft spacing minima at the runways of the airports and the arrival and departure fixes of

bounding the terminal airspace of the metroplex, accounting for uncertainty in the taxi times of the flights and interactions of the multi-airport traffic at the shared arrival and departure fixes. We first present the algorithms for scheduling a single future for flights to comply with inter-aircraft spacing minima at the airports' runways and metroplex' s fixes (Section 4.1) We then present the methodology for selecting the a single scheduling solution from among the set of scheduled futures (Section 4.2).

4.1 Multi-Airport Scheduling For a Single Future

The design of the Phase II arrival and departure scheduler is based loosely on the concepts and capabilities of the Departure Management System (DMS) and Time Based Flow Management (TBFM) automation system currently in use at NY area airports. Departures at JFK are metered by a DMS operated by the PANYNJ. The DMS computes recommended target movement area entry times (TMATs) for individual departure flights to keep movement area taxi times and departure queue lengths manageably small, and sends these TMATs to the airlines and aircraft operators. The airlines and aircraft operators then manage gate pushback times of individual departure flights to adhere to these TMATs. To compute the TMATs, the DMS uses proprietary algorithms that perform ration-by-schedule allocation of departure runway availability to departure flights while maintaining equity among the multiple airlines vying for the utilization of the departure capacity of the airport. Arrivals at JFK are metered by TBFM which assists the ARTCC Traffic Management Coordinators (TMCs) and air traffic controllers with planning and controlling major-airport arrival traffic flows. TBFM enables orderly and metered-to-Terminal Radar Approach Control (TRACON)-capacity delivery of arrival traffic over the arrival-fixes (i.e., for entry into the terminal airspace). TBFM's time-based scheduling engine, called the Dynamic Planner (DP), performs the key time-based arrival scheduling function, and can be adapted to perform multi-airport traffic scheduling. The PROCAST scheduler is largely based on the algorithms and functionality of TBFM and the DP. A key challenge in developing a metroplex integrated arrival and departure scheduler is to manage the use of shared arrival and departure fixes across the metroplex airports.

In this section, we provide an overview of the TBFM Dynamic Planner (Section 4.1.1) and present our emulation of traffic scheduling emulator (Section 4.1.2). We discuss the initialization and operation of the scheduler (Section 4.1.3), methods for spacing and sequencing aircraft (Sections 4.1.4 and 4.1.5, respectively), the process for scheduling aircraft at scheduling points (Section 4.1.7), and how per-aircraft delay to satisfy the schedule is allocated to the flight's planned 4D trajectory (Section 4.1.8).

4.1.1 The Dynamic Planner

TBFM's Dynamic Planner performs the key time-based arrival scheduling function, and can be adapted to perform multi-airport traffic scheduling. DP can handle up to five different airports within a TRACON and treats them as separate entities from the scheduling perspective. A Trajectory Synthesizer (TS) predicts the Estimated Time of Arrival (ETA) at an outer meter arc, the meter-fix, the Final Approach Fix (FAF), and the runway threshold. These scheduling points are collectively called Reference Points. The DP algorithm uses these ETAs to compute de-conflicted Scheduled Time of Arrival (STA) at the meter fix and the runway threshold. The DP first computes STAs to the meter-fix for each arrival-stream, while retaining the FCFS order within the traffic stream (these similar traffic streams are called *stream classes*). The DP may have to delay some aircraft to maintain the mandatory separations between successive arrivals at the meter fix. STAs at the meter fix and nominal fix-to-runway travel times are then used to generate ETAs at the runway. Runway STAs are then computed by de-conflicting the runway ETAs for individual arrival flights with respect to their predecessors to satisfy wake-vortex separation constraints, acceptance rate constraints, and runway occupancy restrictions. The order in which aircraft are chosen for determining their runway STAs was computed by an *Order of Consideration* (OOC) algorithm [W00]. If the delay assigned to a particular aircraft is bigger than the delay absorption capacity (user-specifiable parameter) of the TRACON, then the excess delay was fed back to the ARTCC, and ETAs to the meter fix were updated accordingly. The TRACON delay absorption limit is a user-specifiable parameter of the

DP scheduling algorithm. This parameter is also called the Allowed Maximum Delay Threshold (AMDT). The scheduling process was then repeated for the next flight in the OOC.

4.1.2 Baseline Traffic Scheduler Emulation Functionality

The scheduler developed for this research, referred hereafter as the baseline scheduler, was modeled on the TBFM DP with a reduced number of features. It produces a simultaneous, coupled, multi-airport scheduling solution for both arrivals and departures. While the research focused on the New York metroplex, the baseline scheduler is designed to handle arbitrary metroplex geometry with an arbitrary number of airports, departure fixes, arrival fixes. To simplify the scheduler attributes, boundary acceptance rate constraints were eliminated leaving the core separation problems at the metering fixes, the runways and the departure fixes. Furthermore, wake-vortex separation constraints and runway occupancy restrictions were reduced to a single separation constraint (expressed in time) for both arrivals and departures at both runways and metering fixes. Also, it was assumed that no delay to meet runway or fix capacity limits could be absorbed within the TRACON. All delay feedback had to be absorbed prior to the arrival fixes for arrivals and at the gate for departures. Finally, arrivals were given complete priority over departures at the runway, so that no arrival landing was ever delayed to accommodate a departure takeoff. Figure 3 illustrates the high-level functionality of the baseline scheduler.

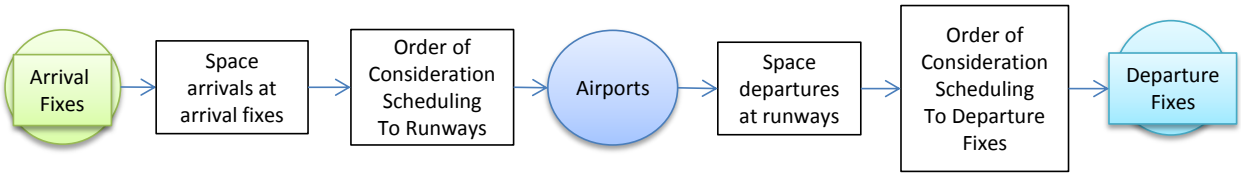


Figure 3. High-level Functional Description of the Baseline Multi-airport Scheduler

There are four main operations for the scheduler: 1) spacing arrivals at the arrival fix, 2) scheduling the arrivals to the runways, 3) spacing the departures at runways (subject to arrival constraints), and 4) scheduling the departures to the departure fixes. The scheduler has only one point of control for metering aircraft. For arrivals, the scheduler can specify a crossing time (STA) at the metering fix, and for departures the scheduler can specify a time for leaving the gate. Once a departure begins taxiing, or an arrival crosses the arrival fix, it is considered frozen and its schedule can no longer be updated.

4.1.3 Scheduler Initialization

There are two main components to initialize before the main scheduling functions can be employed. These are the TRACON model, and the trajectory modeler.

4.1.3.1 TRACON Model

Developing the TRACON model is the initial step for setting up the scheduler. An arbitrary TRACON model is depicted in Figure 4.

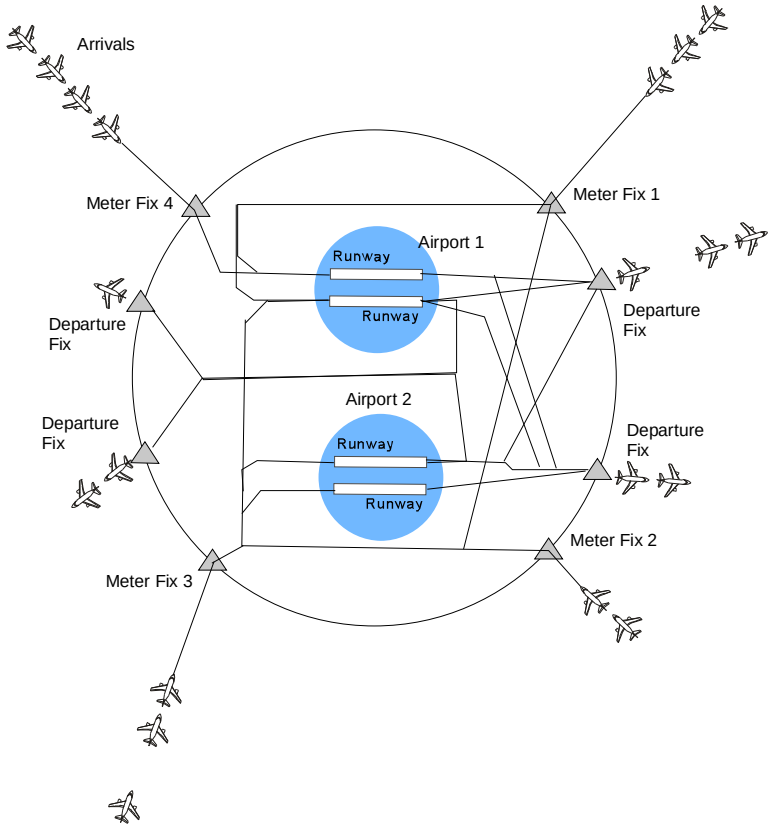


Figure 4. Sketch of an Arbitrary TRACON

A data file defining the TRACON indicates the number of arrival fixes, departure fixes, airports, and runways in the scenario. In this project, the SOSS adaptation data files for JFK, EWR and LGA defined these points. These points serve as both the scheduling points for the model and a means to categorize the aircraft. Each departure and arrival fix has a stream class to further separate classes of aircraft. In the developed model, separate jet and prop stream classes were implemented, however, only the jet stream classes were used due to the nature of the traffic inherent in the FAA traffic schedules used for the project. In the software, each scheduling point (e.g. runway, fix and stream class) is represented with a special array designed to keep track of all the flights assigned to that particular reference point.

4.1.3.2 Trajectory Transit Time Model

Transit time models are needed to obtain the ETA of an aircraft to one of its important scheduling points. Figure 5 illustrates the times needed for PROCAST and the trajectory segments that they represent.

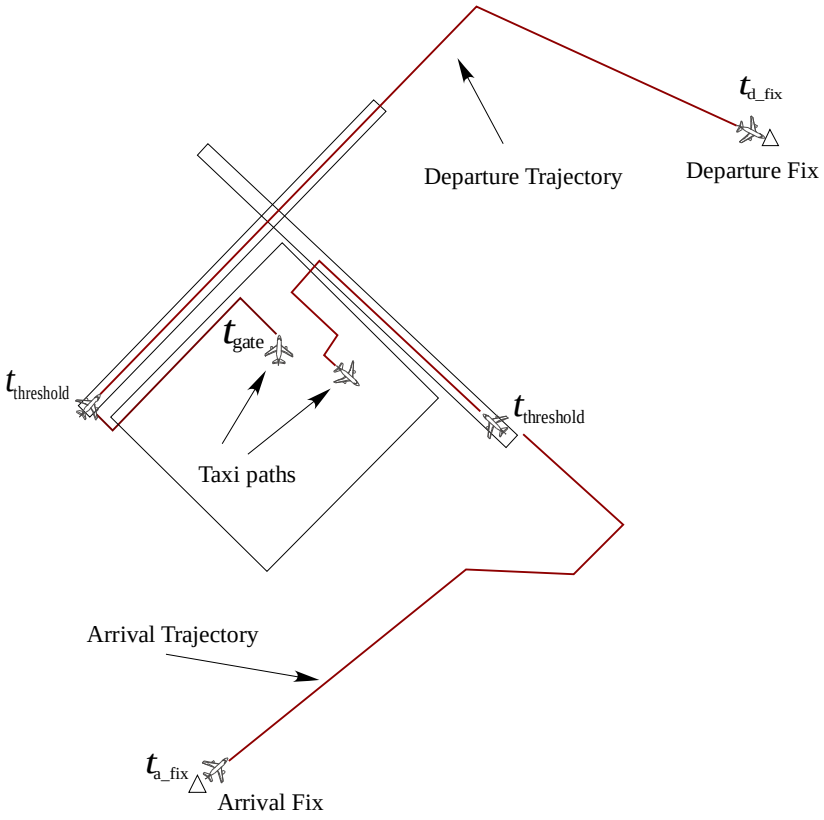


Figure 5. Sketch of Required Transit Times and the Important Points of Interest

For departures, since the only point of control for the scheduler is the gate push time, the time to the runway from the gate is the essential time to capture. For spacing at the departure fix, the transit time to the departure fix is needed.

For arrivals, the time at the metering fix and the time to the runway are the important times. The meter fix crossing time is the point of control that the scheduler has for arrivals, so the scheduler is not concerned with the time required to taxi to the gate.

Traditionally, the ETAs of an aircraft to various Reference Points for scheduling would be provided by a trajectory modeler that predicts the future state of an aircraft using one of several trajectory modeling techniques available. In PROCAST, the ETAs of an aircraft to various Reference Points for scheduling are provided by sampling the probabilistic BN models of transit time or by tabular transit data where no probabilistic model is available, and cascading the successive transit times from an initial time at the first point in the flight's 3D route. In this project, tabular transit data representing "unconstrained" transit of a flight was obtained by running SOSS and allowing all flights to proceed without any interference; that is, with the SOSS Conflict Detection & Resolution (CD&R) functionality turned off.

4.1.4 Scheduler Operation

Once the scheduler is configured with TRACON and flight data, the main scheduler operation begins. The data is received from each of the SOSS processes in the operation environment, and the scheduling cycle is initiated.

The scheduling cycle starts with the arrivals. Our schedulers give complete priority to arrivals over departures, such that no arrival is ever delayed to make room for a departure. With this assumption, a busy schedule of arrivals can fill up the available landing time, causing departures to be scheduled far in

the future. Arrivals are first sorted and spaced at the arrival fixes and then scheduled to their respective runway using an OOC algorithm (see Figure 6).

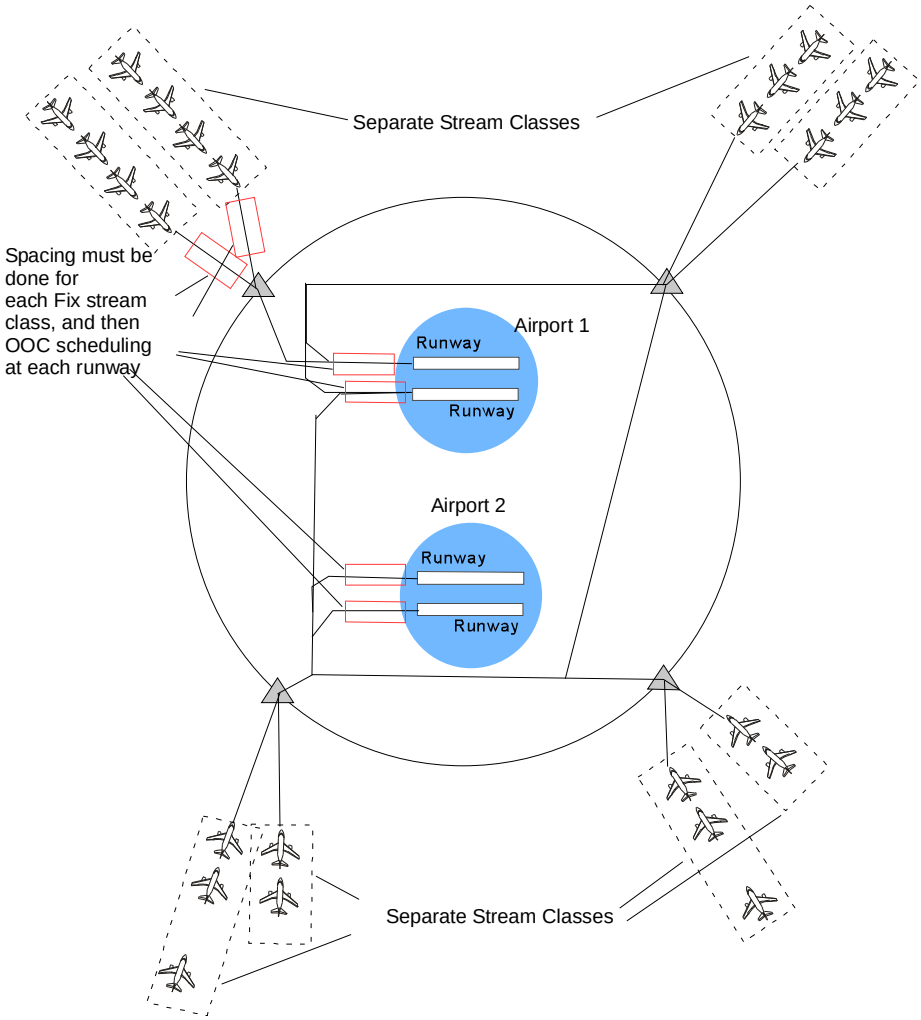


Figure 6. Arrival Spacing and Scheduling within the TRACON

At the arrival fixes, the flights are sorted into their respective stream classes in FCFS order and then each flight is delayed as needed so that the appropriate time spacing exists between all the flights. Next, the OOC algorithm is used to determine the sequence in which the arrival flights should be scheduled at the runway. Using this sequence, each arriving flight is then scheduled to its respective runway. Any delay that is added during the process is fed back to the arrival fix, since no delay is absorbed in the TRACON.

The departure operation is largely the same series of operations, but operated from the runways to the departure fixes. The flights are first spaced at the runways and then an order of consideration algorithm is used to determine the sequence of spacing at the departure fixes. Here, as well, aircraft are sorted into their respective stream classes. The operation is shown in Figure 6. The major distinction between the operations is that runway usage for departures is constrained by the arrival operations. This makes the spacing operations more complex.

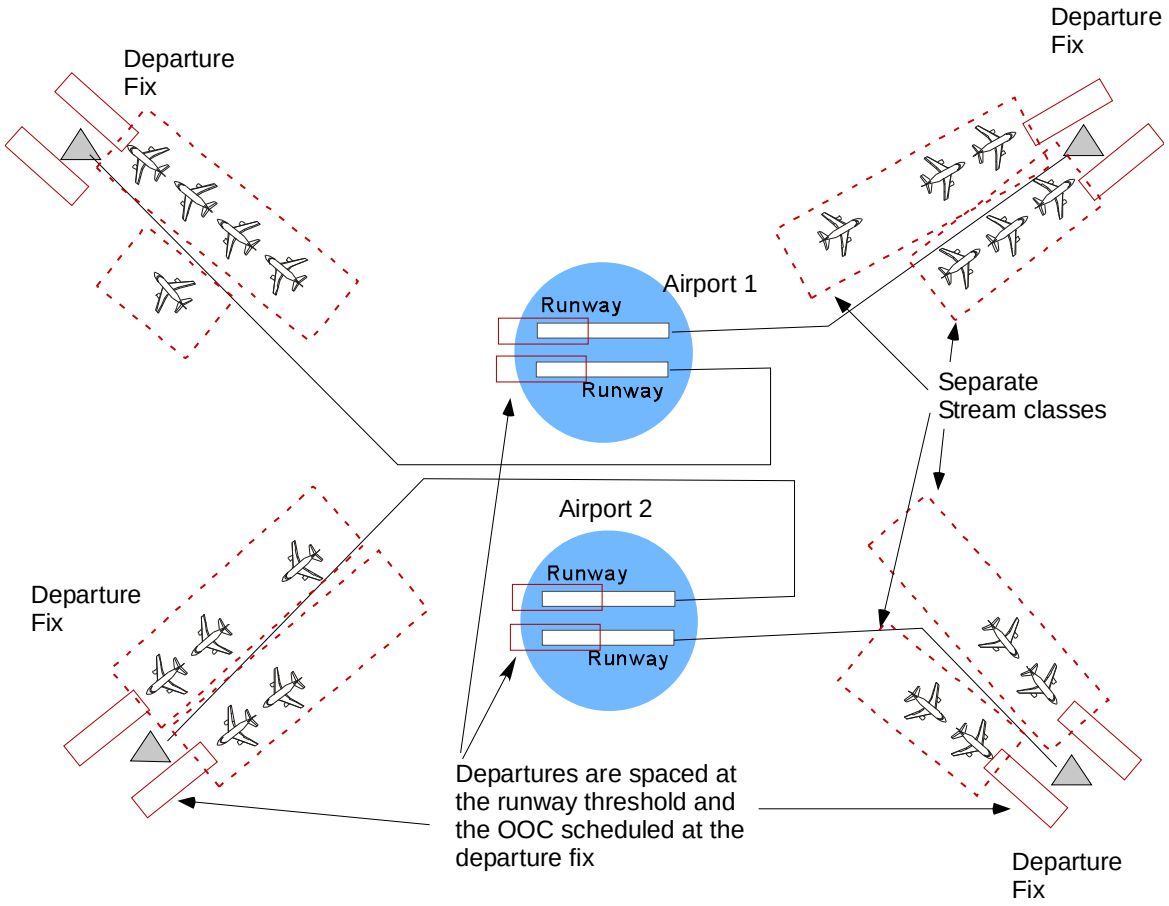


Figure 7. Departure Spacing and Scheduling within the TRACON

The algorithms that perform the spacing and scheduling operations are similar regardless of whether the operation is an arrival or a departure. Within the code, often the identical algorithm can be used for both, just by changing the input parameters. The explicit details on the spacing and sequencing operations are covered in the remaining sub-sections.

4.1.5 Spacing Operations

The general spacing operation is illustrated graphically in Figure 8 and represents how spacing is done for arrivals. First, the subset of flights that correspond to the specific spacing entity (stream class or runway) are collected. Next, the flights are sorted so that their crossing-times are in order.

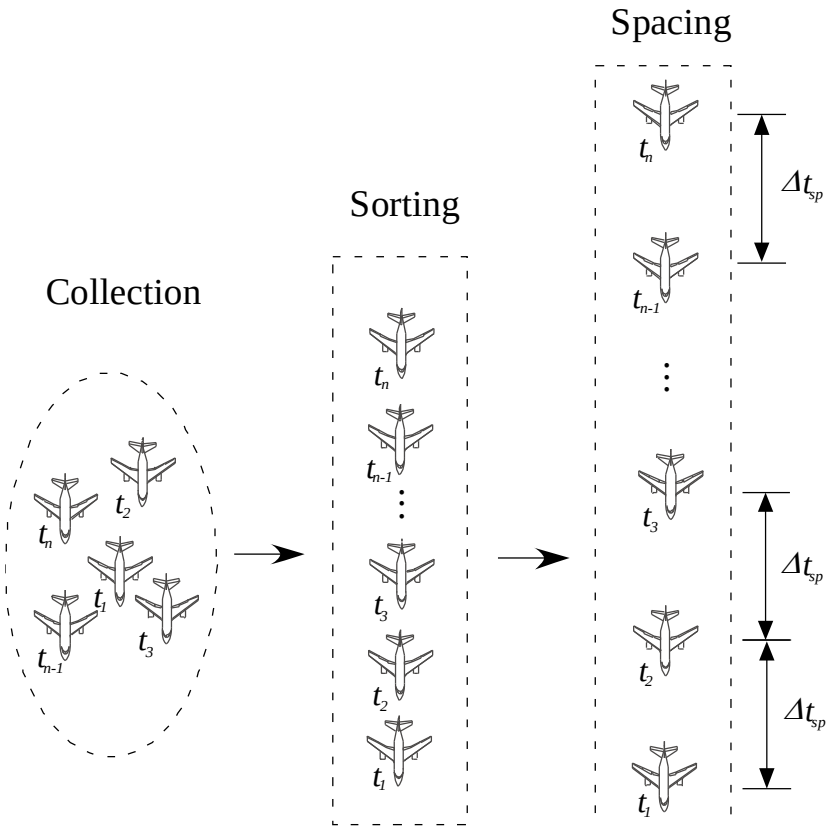


Figure 8. Basic Spacing Functionality

These times are either the fix crossing time for arrivals or the threshold crossing time for departures. Finally delay is added to these flights as needed to insure sufficient spacing in time, Δt_{sp}, is provided. A flow diagram shows how this operation is explicitly performed in Figure 9.

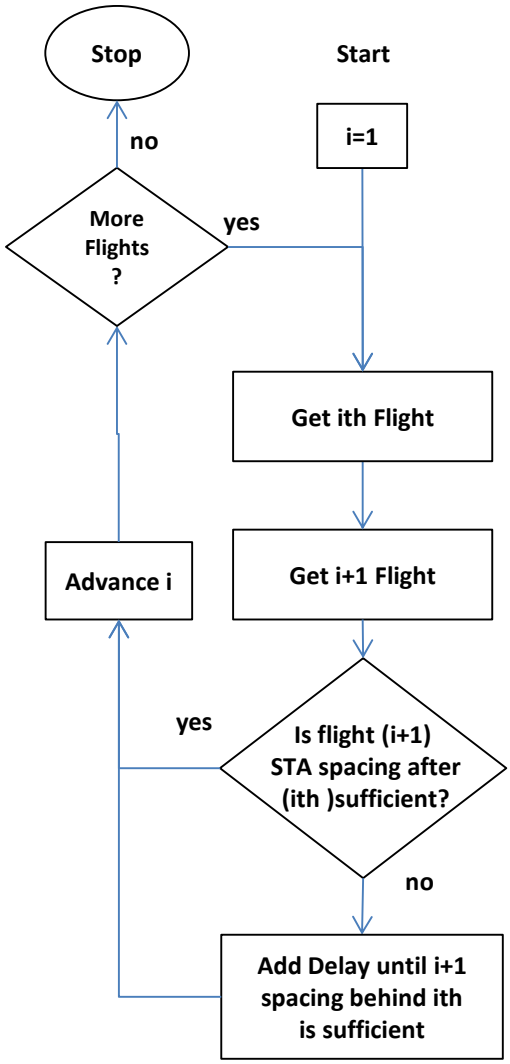


Figure 9. Basic Spacing Flow Diagram

For departures, the operation is complicated by the fact that arrivals are also using the runway and thus serve as a constraint at the runway threshold. To space with constraints, the algorithm maintains a list of the constraint aircraft (e.g., arriving to the same runway) in addition to the aircraft to be spaced. The algorithm spaces the departures among the arrivals as shown in Figure 10. Flow diagrams in Figure 11 show exact method. The left side of the figure shows the basic spacing algorithm with a single function added to check the constraints at the end of each spacing sequence. The constraint-check operation is then shown in the right side of the figure, where location (in time) of the flight to schedule is checked against all the constraints.

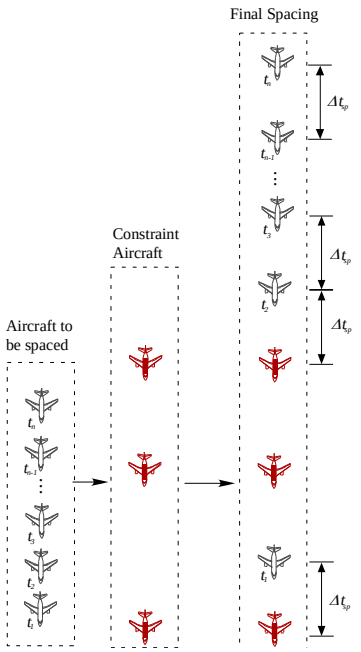


Figure 10. Spacing with Constraints

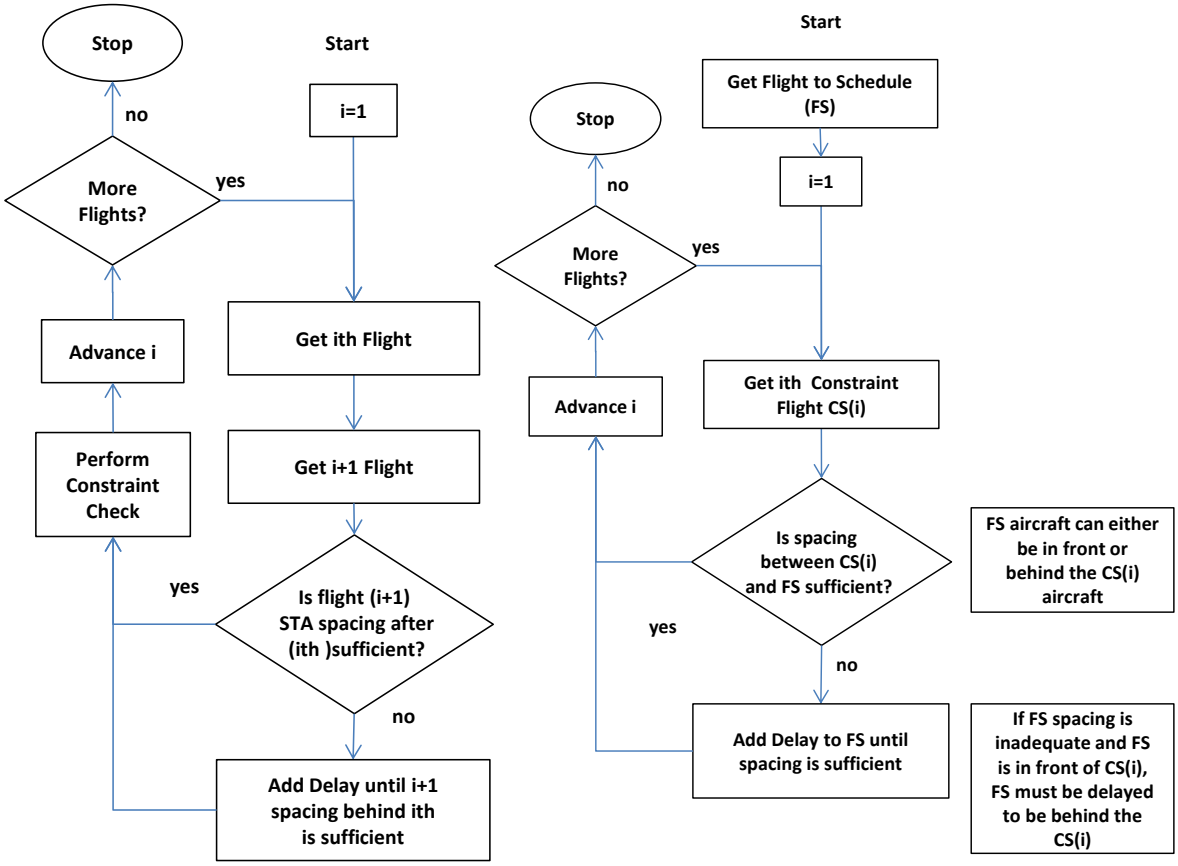


Figure 11. Flow Diagram for Spacing with Constraints (Left) and the Constraint Check Algorithm (Right)

4.1.6 Order Of Consideration

The Order Of Consideration algorithm is used to determine the scheduling sequence of flights from their initial starting locations (i.e. either runway or arrival fix) to their final scheduling points. The identical algorithm is used for both arrivals and departures, so the start and end points are abstracted as the *start points* or the *schedule points*. Two sequences of the OOC algorithm are shown (see Figure 12 and Figure 13) to illustrate the process. In Figure 12, the start points are shown at the top with their spaced sequences of flights that all need to be scheduled to the schedule points (circles) below. First, the earliest flights (with respect to its respective start point (e.g. fix)) from each start point are chosen and placed in a special array, the scheduling bin, for scheduling consideration.

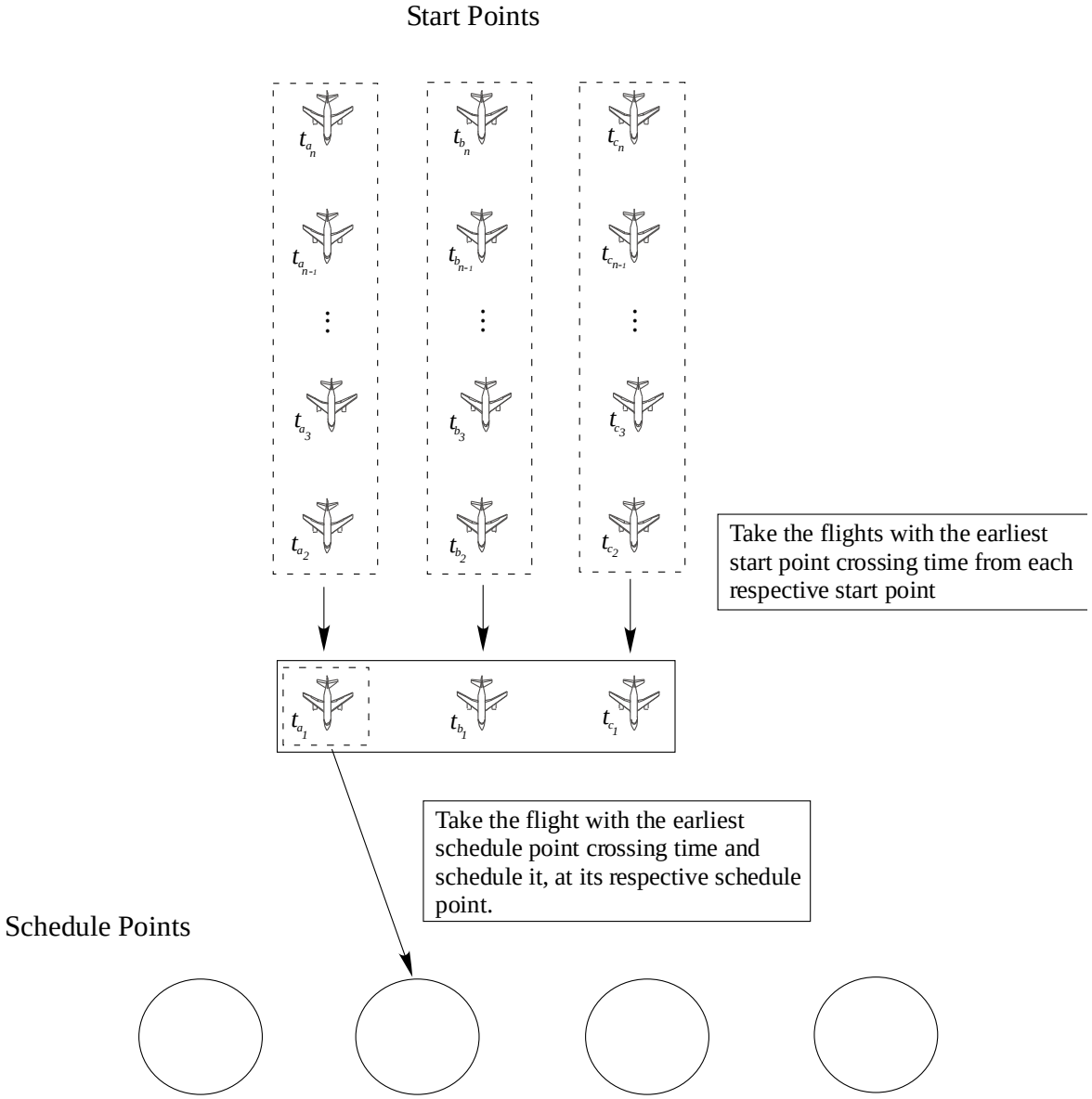


Figure 12. Initial Order of Consideration Sequences

Of the flights in the scheduling bin, the time that each flight is estimated to reach its scheduling point (e.g. runway) is evaluated. Of the scheduling bin flights, the flight that reaches its respective scheduling point first is selected for scheduling. It is an important detail to note that the flights in the scheduling bin are likely destined for different scheduling points. Once an aircraft is selected for scheduling, it is scheduled

using an operation that is totally independent of the OOC algorithm. On the next sequence of the algorithm, the start point which previously provided the scheduled aircraft provides its next earliest flight to fill the gap in the schedule bin (see Figure 13), and the process is repeated. Figure 14 captures the process in a flow diagram.

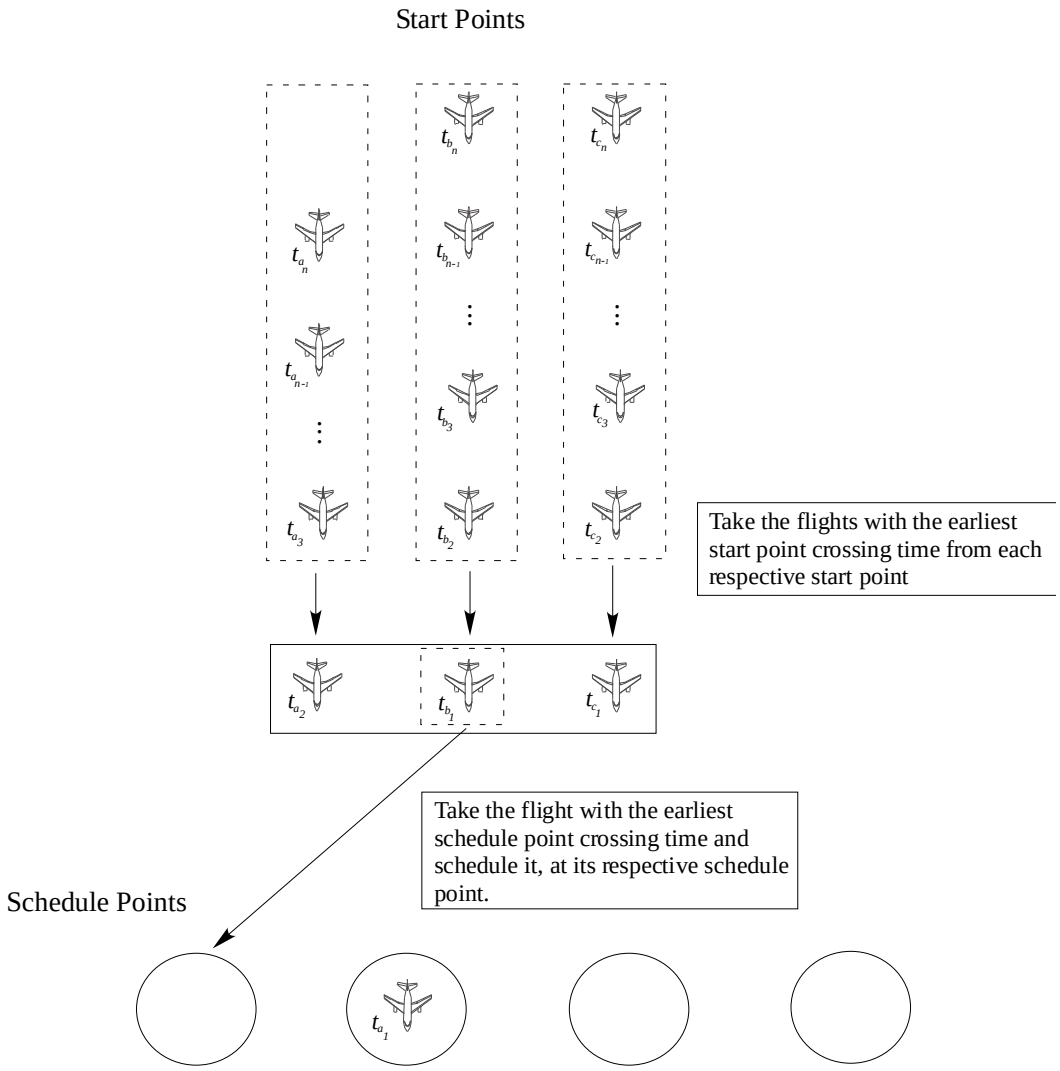


Figure 13. Successive Order of Consideration Sequences

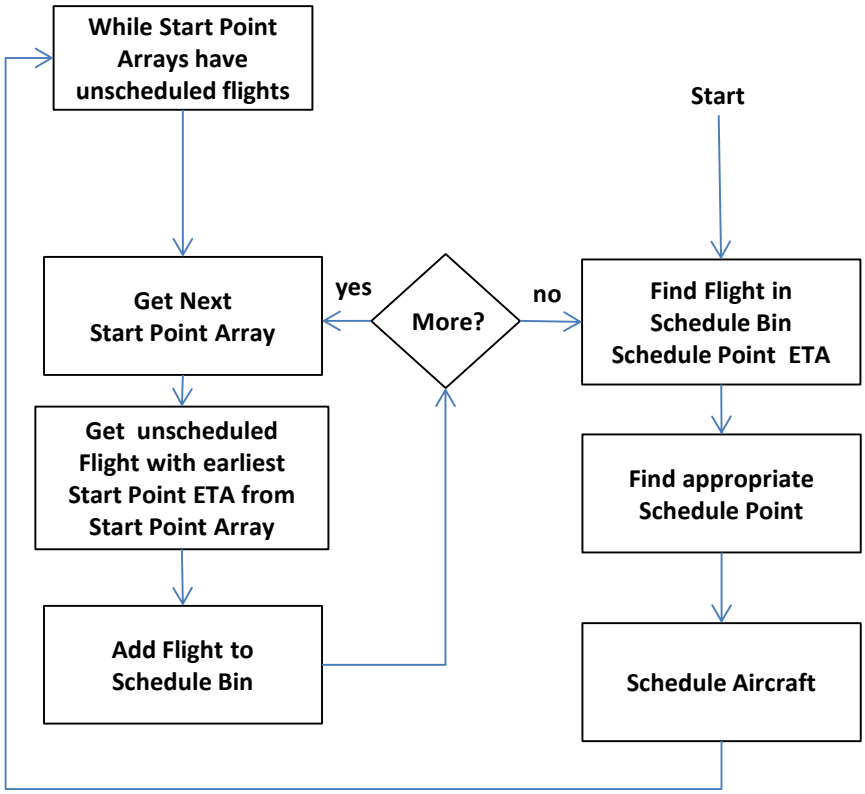


Figure 14. Successive Order of Consideration Sequences

4.1.7 Scheduling at Scheduling Points

Scheduling flights at the scheduling points involves finding an appropriate time slot for the currently schedule flight amongst all the flights previously scheduled at the particular schedule point. Nominally, this involves just adding the flight to the back of the list and adding a sufficient amount of delay to make sure that it is appropriately spaced behind the last scheduled flight. The nominal case is illustrated in Figure 15. This is the most likely scenario, particularly with arrivals. However, it is possible for the scheduling scenario to be such that a particular flight later in the scheduling sequence (as determined by the OOC) will arrive at the scheduling point earlier than the previously scheduled flights. This problem is exacerbated by a multiple aircraft scenario where transit times to runways may differ substantially due to fix-runway proximity differences. Therefore, allowances are made for scheduling a current flight in between the other flights. This is particularly true of departures at departure fixes, where there can be large time segments in between prior departures. Figure 16 illustrates this rarer case. While rescheduling prior flights in the schedule sequence is not permitted, if sufficient space exists between flights such that the current flight can fit, that space is made available to the flight. In this case, spacing must be checked between the leading and the trailing flights. A flow diagram that captures the actual algorithm is shown in Figure 17. In the flow diagram the current flight is checked against all the prior aircraft until an appropriate slot is found. Successive amounts of delay are added until the flight finds either a slot in between flights, or is put at the back of the sequence.

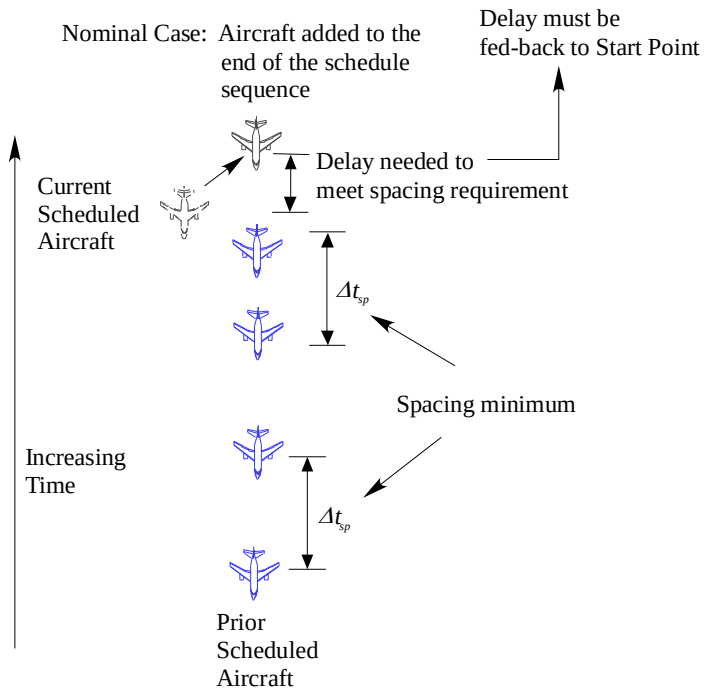


Figure 15. Nominal Scheduling Case

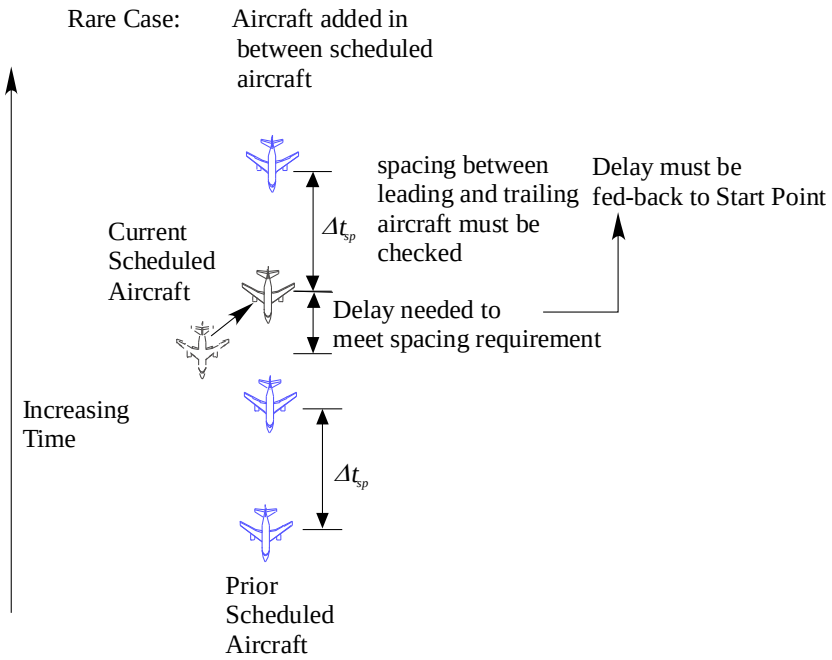


Figure 16. Rarer Scheduling Case

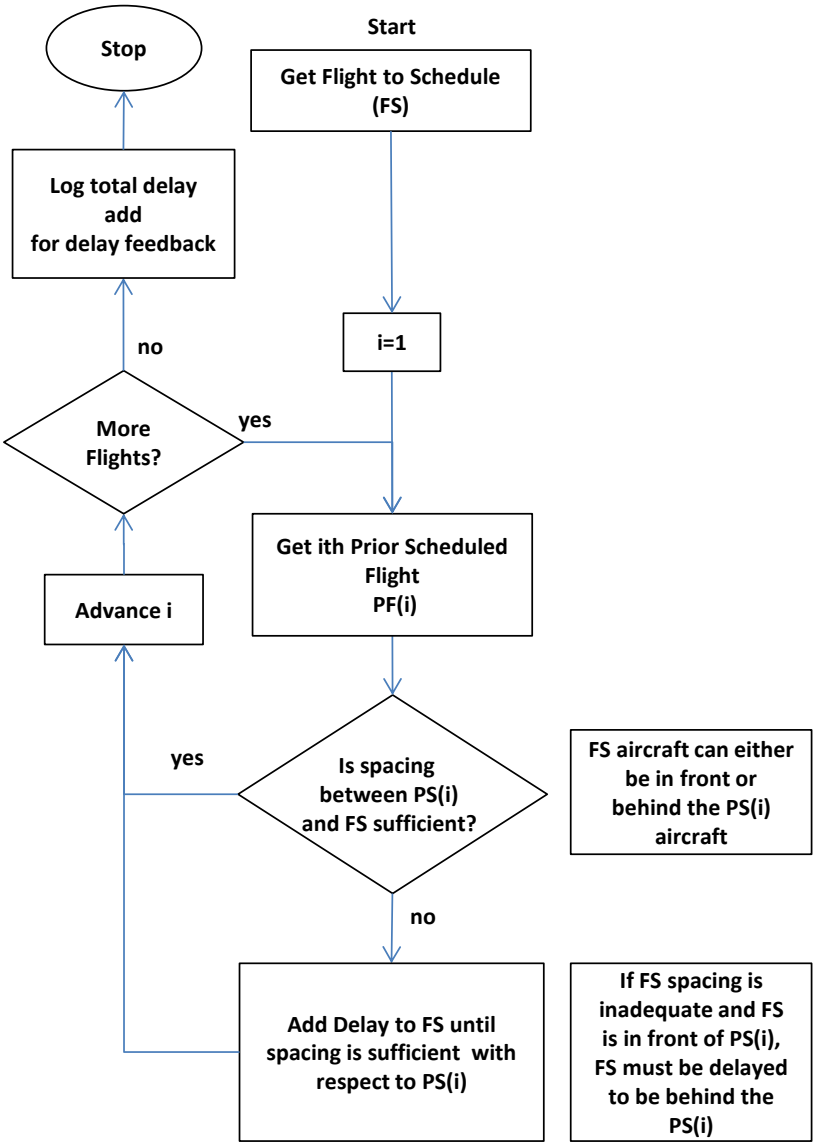


Figure 17. Flow Diagram for Scheduling

4.1.8 Delay Feedback

Delay feedback, the delay that is accumulated at the scheduling point that must be fed back to the start point, complicates the algorithm because it requires a respacing of the aircraft at the start point if the delay is enough to create a spacing conflict. Managing delay feedback for arrivals is the easiest. This involves re-running the initial spacing algorithm for the particular arrival-fix stream class which contains the flight. Any time an arrival is delayed at the runway, the spacing algorithm for the stream class must be run again. The algorithm does this immediately upon delaying an aircraft so that any delay that might ripple through the stream class is accounted for in the successive scheduling operations. This operation need only be performed once and only impacts aircraft downstream (in time) from the current flight under consideration.

However, for departures, delay feedback is more problematic. This is due to the need to not only space with departures, but also space with the arrival constraints on the runway. Therefore, it is possible that the current flight might have been put in conflict with an arrival due to the schedule process at the scheduling

point (departure fix in the case of departures). To resolve this conflict, delay beyond what is required at the scheduling point is needed. Of course, any delay added at the runway to resolve a conflict with arrivals may create a conflict in the departure stream.

To resolve this problem both the schedule point algorithm and the constrained spacing algorithm are run in an iterative fashion, each adding delay until no conflicts are found. Figure 18 shows the flow diagram. If delay is added during either of the operations, then both need to be re-run to insure a conflict free solution. Once a sequence is run with no additional delay added, the algorithm exits.

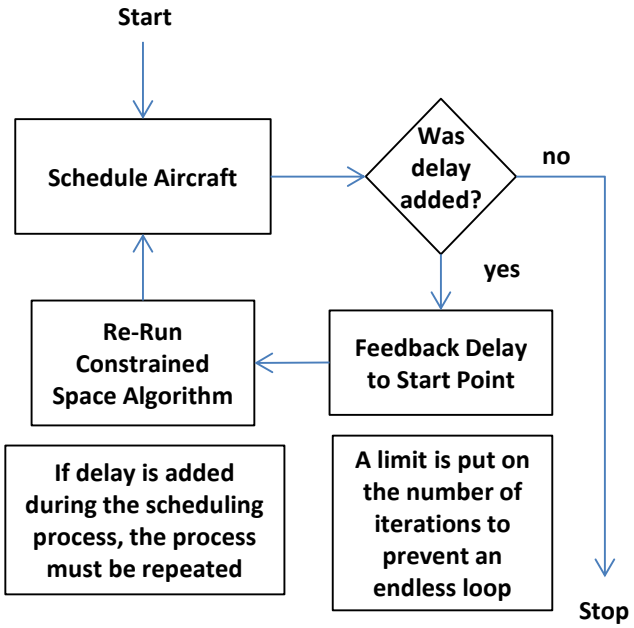


Figure 18. The Iterative Solution for Simultaneous Solution of Runway and Departure Fix Conflicts

4.2 Schedule Specification From Multiple Scheduled Futures

PROCAST scheduling works as follows. The BN models are used to generate a set of, say, 100 futures for the aircraft waiting at their gates, ready to depart. For each future, multi-airport scheduling is performed in a deterministic fashion to specify, for departures, the schedule of release times of the aircraft waiting at their gates, ready to depart, and their inherent runway times. A deterministic scheduler schedules the fix, runway and pushback times of the departures in each future, and does this for each future, resulting in N distinct sequences and schedules, one for each future. Such scheduling is performed for each of the 100 futures predicted by the BN, resulting in a distribution of 100 possible release times for each of the aircraft waiting at their gates, ready to depart. The scheduled pushback times for departures are selected from among the pushback times across all the futures.

At each scheduling cycle, the PROCAST scheduler receives the current state of all active and pending flights (arrivals and departures) in the planning time horizon. The set of flight states is extended to a *future* by predicting the future 4D trajectory of each flight based on its 3D path and a probabilistic model of transit time. In the present work, departure taxi time was modeled probabilistically, but arrival taxi time and departure and arrival transit time in the terminal airspace were predicted as "unimpeded" transit times. The probabilistic BN model predicts a probability distribution of arrival times for each departing flight to points along its 3D route. A future is constructed by sampling from this arrival time distribution to construct a predicted 4D trajectory for each departure. By repeating this procedure, the set of current flight states is extended to a number of distinct futures, each one describing, plausibly, what might happen if the traffic was be allowed to run its course without further intervention.

At this point, PROCAST schedules the arrivals and departures in each future. To estimate times of arrival for flights to the scheduling points, PROCAST uses the predicted transit times of the flights (inherent in their predicted 4D trajectories) in that future. The traffic scheduler assigns release time to each flight as a gate pushback time for each departure that is at the gate, an arrival fix crossing time for each arrival that has not passed the arrival fix, and all near future flights. Because the predicted transit times for a given (departing) flight differ from one future to the next, the release times the traffic scheduler assigns to flights in each future to meet constraints at the runway and the fixes will be different. All of these release times are in the future, i.e., later than the current simulated moment.

Once scheduling has been performed for each of the futures, probabilistic analysis is used to choose the "statistically-best" traffic schedule. PROCAST currently does this by selecting the "statistically best" scheduled future among the set of scheduled futures, using a method applied in Phase I of the project. In this method, a particular scheduled future is selected by 1) successively applying a data outlier filtering technique to the scheduled futures, 2) arbitrarily selecting a scheduled future from among the remaining scheduled futures. Consider the delay of a flight in a given future is the difference between its final scheduled an initial estimated gate pushback or fix crossing times. The outlier filtering technique eliminates any futures which, for a given flight, prescribe delays to that flight which lie outside the range of the mean +/- 1 standard deviation of the delays for that flight. The outlier filtering technique is applied for each successive flight to the current set of scheduled futures, where flights are sequenced in order of greatest to least mean delay. As the scheduled futures are filtered from the set, the mean and standard deviation for the flight are recomputed. The following pseudo-code gives the algorithm:

Method selectBest(futures) **returns** future

1. **Let** delays(f, j) be a matrix; the delay of flight j in future f.
2. **Let** meanDelays(j) be the mean delay of flight j across all futures.
3. **Let** stdDelays(j) be the standard deviation of the delays for flight j across all futures
4. **Sort** meanDelays in decreasing order
5. **Sort** stdDelays so that the flights are in the same order as in meanDelays
6. **Sort** delays(f, j) so that the flight columns are in the same order as in meanDelays
7. **For each** flight J (as ordered in meanDelays)
    a. **Compute** the range meanDelays(J) ± stdDelays(J)
    b. **For each** future F
        i. **If** delays(F, J) is outside of that range, **then** mark future F for deletion
    c. **If** all futures are ***marked*** for deletion, **then** exit the loop
    d. **Delete** all of the marked futures (i.e., down-select)
        i. **Remove** from the set of futures
        ii. **Remove** the corresponding rows of delays matrix
    e. **If** exactly one future remains, **then** exit the loop
    f. **Recompute** the meanDelays and stdDelays from the remaining futures. N.B., this can be restricted to just flight J+1.
8. **Return** an arbitrary future from the set of remaining futures

This procedure will approximately select a future in which delays for each flight are near the mean for that flight. In doing so, it will favor flights whose mean delay is greatest. However, note that re-computing the means and standard deviations after each down-select (step 7f) complicates this explanation.

By applying this methodology, PROCAST selects a scheduled future for the flights in the planning time horizon. The scheduled future specifies a *planned* 4D trajectory for each flight which prescribes gate pushback and takeoff times for each departure or arrival fix crossing and landing times for each arrival. PROCAST implements this set of control actions, i.e., the collection of landing and fix crossing times for arrivals and takeoff and gate pushback times for departures to manage the traffic.

5 Bayesian Network Transit Time Modeling

This section details the development of probabilistic models of departure taxi time using Bayesian networks for use in PROCAST's multi-airport scheduling, beginning with the model of taxi time developed in Phase I. Results reported in this section are primarily the work of the Intelligent and High-Performing Systems Lab at Carnegie Mellon University, led by Dr. Ole J. Mengshoel[1], and with participation from Aniruddha Basak, Priya Sundararajan, Erika Menezes, and Vinodh Paramesh.

Our test system is a simulation of the New York metroplex. This consists of three linked copies of SOSS, individually simulating the ground and air traffic at JFK, EWR, and LGA airports (note that our Phase I work extended SOSS to simulate air traffic in the TRACON). Although the airport simulations are separate instances of SOSS, the airports modeled in SOSS share common arrival and departure fixes, and the SOSS simulations of the airports are synchronized at each scheduling cycle. At each scheduling cycle, the arrivals and departures of the three airports in the metroplex are scheduled in a unified manner, thus enabling de-confliction at the shared arrival and departure fixes.

As in Phase I, the required BN models need to predict probability *distributions* for the taxi time of individual departures. These distributions depend on the specifics of the departing flight and on ground traffic conditions at the departure airport during taxiing. We expect the taxi times at each airport to be relatively independent of conditions at the other airports, and to have different random variables (BN nodes) which influence the taxi times. Thus, for simplicity, we developed a separate taxi time model for each airport.

Our general approach, followed in both phases of the project, is to treat the taxi time, aspects of the taxiing flight, and the traffic levels at specific points on the tarmac as random variables in a joint distribution. We use machine learning to create, from data, an approximation to this joint distribution, in the form of a BN. During scheduling, we predict the posterior distribution of the taxi time, given the features of the flight, and approximations to the traffic levels. The BN allows us to compute this posterior probability distribution conveniently and efficiently. For scheduling purposes, we sample from the posterior distribution repeatedly, in order to construct a set of plausible future scenarios, based on the current state, and with appropriate probabilities.

Our random variables include the taxi time from gate to runway, as well as from gate to two intermediate points along the trajectory of an aircraft on the airport surface. If a flight is already taxiing at the moment when we are scheduling, and, if it has already passed either of these two points along its taxi path, then we will supply the actual time taken to reach the point(s) as evidence in order to produce a more refined prediction of the gate-to-runway time.

We began, in Phase II, by revisiting the taxi time model for JFK from Phase I to understand its performance, and, to see if it could be improved. We reconsidered certain design decisions made during Phase I, such as treating taxi time as a discrete variable, and using stage-wise sampling of the posterior distribution. We also uncovered and characterized a particular aspect of the Phase I model that sharply limited its accuracy: zero probability values in Conditional Probability Tables (CPTs). We explored a number of approaches to handling, mitigating, or removing these in the Phase II work reported on here.

[1] See https://www.cmu.edu/silicon-valley/faculty-staff/mengshoel-ole.html

To compare different modeling approaches in a principled and systematic manner, one of our first tasks was to design and build an evaluation framework. This is software that uses a data set (of taxi times and other variables) to both train the BN model, and to compute metrics and display graphs of its prediction performance. Our initial evaluation framework used the test and training data from Phase I. This framework was only applicable to models of taxi time at JFK. Later, the framework was adapted to use a much more extensive and elaborate data set, covering all three airports.

We also note that the Phase I BN model, which is specific to the JFK airport, refers to unique locations on the surface of JFK. The points include spots, taxiway merge points, taxiway crossing points and runways. These points, and the structure of the BN, were arrived at by trial-and-error, with input from subject-matter experts familiar with the airport. In Phase II, we sought to generalize this model so that other airports (and other airport configurations) could be straightforwardly handled within the same generalization. To be clear: the generalization involves the choice of random variables and structure in the BN. The parameters of the network are still derived from our data set by machine learning.

As part of this generalization, we experimented with BN *structure learning*, in which the structure of the network, for a given set of random variables, is constructed from data using machine learning techniques. Moreover, we created a generalized set of features which are broader, much larger than in Phase I, and applicable for any airport.

Putting these elements together, we constructed predictive models of taxi time for all three airports. For JFK, a comparison of the Phase I model and the new data-driven model developed in Phase II showed a marked improvement in predictive power for the new model. In this comparison, both models were trained and tested with the Phase II data set. The models resulting from this data-driven approach were then used in our scheduling experiments, as described in Section 5.2.7.

The following subsections discuss the details of these accomplishments. Section 5.1 covers the evaluation and improvement of the Phase I taxi time model. Section 5.2 covers the generalization of the model, use of the more extensive Phase II data set, and the creation of data-driven models for taxi time at all three major airports of the New York metroplex.

5.1 Enhancements to Phase I Bayesian Network Taxi Time Models for JFK

This section presents our Phase II methods and results of enhancing the Bayesian network models of taxi time for JFK that were developed in Phase I of the project. Here, we have developed a systematic and principled method to developing and evaluating transit time models for airports. The approach is based on an airport-specific Bayesian network transit time model, for JFK, developed in previous work.

We begin with a description of the data set from Phase I that we used initially (Section 5.1.1). We then discuss the initial evaluation framework and its components (Section 5.1.2). We also describe an algorithm that outlines the training, testing and evaluation of the framework in more detail. We then explain improvements relative to Phase I. First, we fixed the bug in an open source Bayes Net Toolbox software package (see Section 5.1.3) for MATLAB. Second, we tried a simultaneous sampling approach of generating futures which was much faster compared to the stage-wise sampling approach (Section 5.1.4). Third, we investigated different approaches to fix the problem of zero probability values (Section 5.1.5).

5.1.1 Raw Phase I Data

The initial dataset, adopted from Phase I, is comprised of the output of 501 SOSS simulations of JFK. In each of these runs, the SOSS simulation is operating deterministically (with no pseudo-randomness), and with the SSOS CD&R functionality turned on. The simulations are all for the same six hour schedule of flights, where each flight has a fixed 3D path. The simulations differ because the gate pushback times or landing times of each flight have been perturbed pseudo-randomly, external to the simulation, according to a simple model of gate pushback time uncertainty. Within the simulation, the individual taxiing flights

interact with each other via common points in their 3D routes and the SOSS CD&R mechanism (i.e., they are not allowed to crash into one another). The details of these interactions are sensitive to initial conditions, and thus, taxi times vary from one simulation run to the next in a manner that would be difficult to predict.

5.1.1.1 4D Trajectories From SOSS

The output of the SOSS simulation runs (captured in the *ACSchedule* files) describes the progress of each flight along its 3D path in a link-node (a directed graph) model of the airport. The file contains the 4D trajectory of each flight consisting of the sequence of nodes in its 3D path and the time of arrival to and release from each node in its 3D path. Thus, each *ACSchedule* file describes a set of taxiing flights that are all mutually interacting and thereby mutually impeding their progress along the taxiways.

The 3D route of each departing flight in the Phase I SOSS simulation data for JFK includes four distinct nodes the flight passes through: the *gate*, *spot*, *merge*, and *runway* nodes. The *gate* node is the first node; it is where the departure originates in the simulation. In the Phase I SOSS simulation data, the *runway* node is the last node; it is the point on the runway from which the aircraft begins its acceleration for take-off. In the Phase II SOSS simulation data, the 3D route of each departure includes an additional node after the runway, the *departure fix* node; it is the point where the departure has completed the significant portion of its climb towards cruise, and exits the TRACON airspace.

A *spot* is a point on the airport surface where a flight passes between the ramp area (controlled by the airlines) and the movement area (controlled by the air traffic controllers). The SOSS adaptation for an airport identifies a subset of nodes as *spot nodes*. Each departing flight will pass through *one* of these spot nodes.

All of the departures using a given runway will approach that runway through the same final sequence of nodes. The first of these common nodes is called the *merge node*, because the trajectories of departing aircraft all merge into a single aircraft stream at this point. In the airport configurations that we used, only one runway is used for departures at each airport. Thus, in our data, we have a unique merge node at each airport.

5.1.2 Evaluation Framework

We create an evaluation framework, shown in Figure 19, to enable the testing of BNs or any other machine learning models. The evaluation framework also helps to refine parameters. As shown in Figure 19, the framework is a pipeline, beginning with raw data from SOSS airport simulations, and ending with performance metrics of the particular model under test. Along the way, the raw data (*ACSchedules*) is processed to extract the departing flights and the relevant features (the "Table of Factors" or TOF), it is split into separate training and testing subsets, and it is optionally discretized (or "binned") if discrete random variables are used in the BN. The parameters of the model (e.g., the Bayesian network) are derived from the training data, using the appropriate form of machine learning. That model is then used to predict the transit taxi times of the flights in the test data. Comparison of these predictions with the actual transit times in the test data, results in performance metrics results.

As indicated in the Figure 19, we have kept the data from each spot (i.e., departing flights that passed through each spot) separate. This enables us to train and test a separate model for each spot, as was done in Phase I. It also allows us to focus on the model and data from just one spot for a more detailed examination.

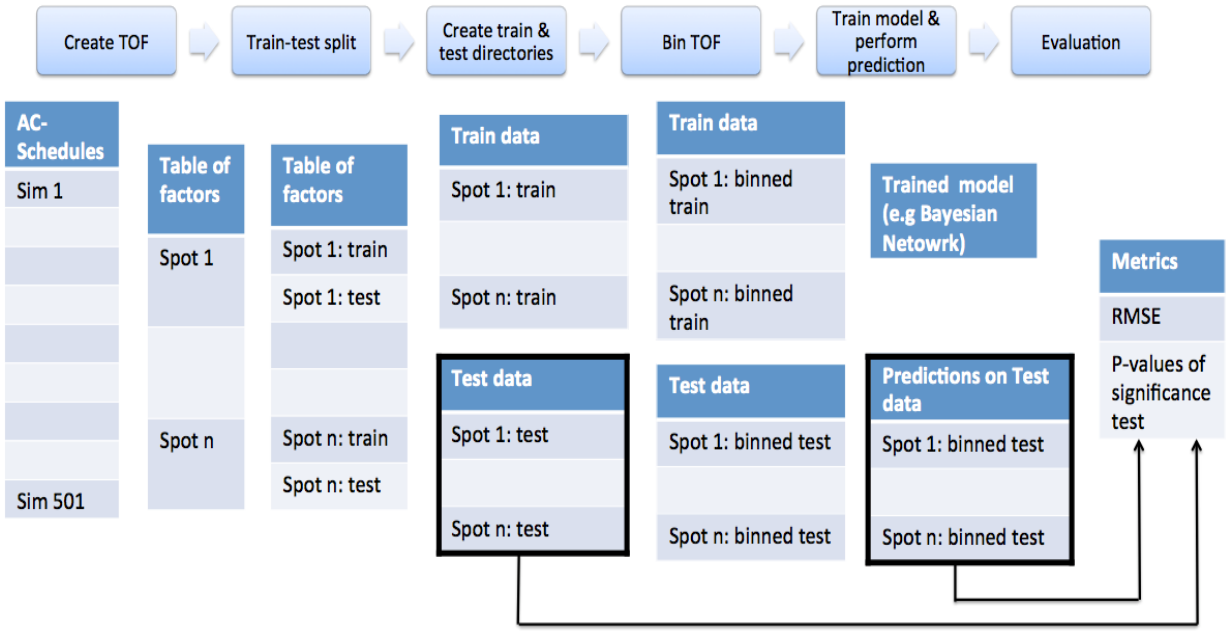


Figure 19. JFK Evaluation Framework

For example, for spot DA015, we show the P-distribution of the runway arrival time in Figure 20. The P-distribution is generated by comparing the actual marginal distribution generated using TOF data (see Figure 24) and the predicted marginal distribution in the trained BN model (see Figure 25). If the P-values are higher, then the two distributions are more similar as shown in Figure 20.

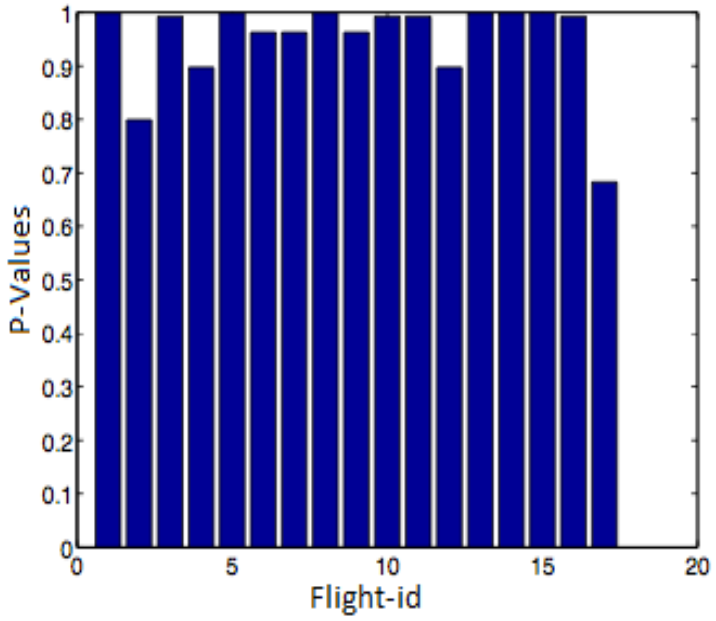


Figure 20. P-Distribution of runway arrival time. The original distribution is shown in Figure 23 and predicted distribution is shown in Figure 25.

5.1.2.1 Creating Table of Factors

The factors relevant to the taxi time of a given departing flight include aspects of that flight, as well as descriptors of contemporaneous traffic. For example, the gate that a flight leaves from is a property of that flight. In contrast, the *simultaneous gate releases* is a count of the number of flights departing from nearby gates in a small window of time surrounding the pushback time of the given flight. The simultaneous gate releases count depends on *other* flights, but taken from the same simulation run. The *Table of Factors* is a compilation, for each flight, of selected aspects of the flight, as well as of other traffic metrics that have been pre-computed using flights from the same simulation run, and at the appropriate simulation time. Once we have created the Table of Factors, we can treat each flight as an individual data point, without additional reference to the simulation run in which it occurred or to other flights. In later sections we will often refer to these factors as *features*, to be consistent with the machine learning literature.

We began with the factors or features from Phase I. These include such fixed aspects of the flight as its gate and spot (i.e., the nodes in the link-node model corresponding to the gate and the spot, for that flight). We refer to this combination of node sequence and times as the *trajectory*. Our data set from the 501 SOSS simulations contains 501 instances of each simulated flight. For each flight, its 501 instances have the same 3D path (as a sequence of nodes in the SOSS link-node model of the airport), but differ in their times at the nodes along that path, thus representing the alternative 4D trajectories of the flight. Some features are extracted from the timing along the flight's 4D trajectory, such as the gate time (*actPushbackTime*), spot time (*actSpotArrTime*), merge time (*actMergeNodeArrTime*), and runway time (*actRwyTime*).

We adapted software from Phase I to extract these features from the data. Although the process is lengthy (about 1 hour of compute time) the result can be cached and reused. Note that the entire data set must be processed, even if only a subset is used for some test, because, for discrete random variables, we must learn the full range of values that are present in the data in order to set the sizes of corresponding BN nodes.

The structure of the data is illustrated in Table 1. The leftmost column names the extracted fields including the call sign, gate, spot and arrival times at various nodes. The call sign is used to identify unique flights. In this example, spot *VA_001* has 2 unique flights namely UAL23 and UAL53. For every *ACSchedule* txt file processed, representing one simulation run, a column is appended to the *table_of_factors_data* structure for each unique flight for that spot. In this case, for spot *VA_001* for the two *ACSchedule* files were processed, four columns are added two for each of the unique flights. The TOF for this spot, for our full data set (for Phase I data) will have 1002 columns; 501 for each of these two flights.

Table 1. Table of Factors Data Structure for Phase I

	1	2	3	4
Callsign	UAL23	UAL53	UAL23	UAL53
Gate	Gate_07_008	Gate_07_008	Gate_07_008	Gate_07_008
Spot	VA_001	VA_001	VA_001	VA_001
…	…	…	…	…
actSpotArrTime	1875.80	19853.80	1949.80	20169.80
actMergeNodeArrTime	2119.64	20097.64	2193.64	20413.64
actRwyRelTime	2353	20354	2426.50	20646.50

The fields that correspond to *actSpotArrTime*, *actMergeNodeArrTime* and *actRwyRelTime* (values of last three fields as shown in Table 1) are time values denominated in seconds from the beginning of the simulation. These have not been discretized (or "binned") in order to allow for experimentation with different bin sizes and comparison of actual and predicted time values. Note that for training the BN engine and for prediction the random variables (of the same names as the temporal variables in TOF) will all be computed relative to the gate pushback time. In particular, in Phase I, and in our current work assessing the Phase I Bayesian network, the gate time random variable (*actPushbackTime*) is always zero.

5.1.2.2 Splitting Data into Training and Test Sets

It is a common practice in machine learning experiments to split the data set pseudo-randomly into training and testing subsets. This is done to force the learning model to generalize to new previously seen and new instances. This method prevents overfitting; that is, models performing very well in the training phase and poorly during test phase.

For this reason, once the table of factors has been created, it is split into two sets, training and testing, in a ratio of 6:4. The split is carried out freshly for each test or experiment.

To split the *table_of_factors* data structure into these two sets, care should be taken to ensure that all unique flights are present in both sets. This is ensured by using the flight ids that is stored in the call sign field of the *table_of_factors*. Referring to Table 1, columns 1 and 2 belong to one simulation instance and will be extracted together into one of the two sets. Similarly columns 3 and 4 belong to another unique simulation and will be extracted into of the two sets.

This splitting of the *table_of_factors* is carried out for each spot to create train and test sets and stored in train and test directories respectively.

5.1.2.3 Training BN and Performing Predictions

Training a BN typically involves two steps: 1) creating a BN structure, and 2) learning the BN parameters. The Phase I BN structure is based on the knowledge of one or more subject matter experts. A snapshot of a small part of the JFK airport, in the form of a node-link diagram, is shown in the left side of Figure 21. Some important junction nodes that are associated with our Phase I random variables are indicated in red. The random variables in the model (i.e., the nodes of the BN) include metrics of the traffic levels at these important junctions in addition to the flight's time of arrival at these nodes. The flight's trajectory along these nodes in the airport snapshot is used to create plausible dependencies of the random variables in the BN as shown in the right side of Figure 21. For example, it is entirely reasonable that traffic at node F010 might affect the transit time of some flights (such as the one indicated in green) from gate to merge node.

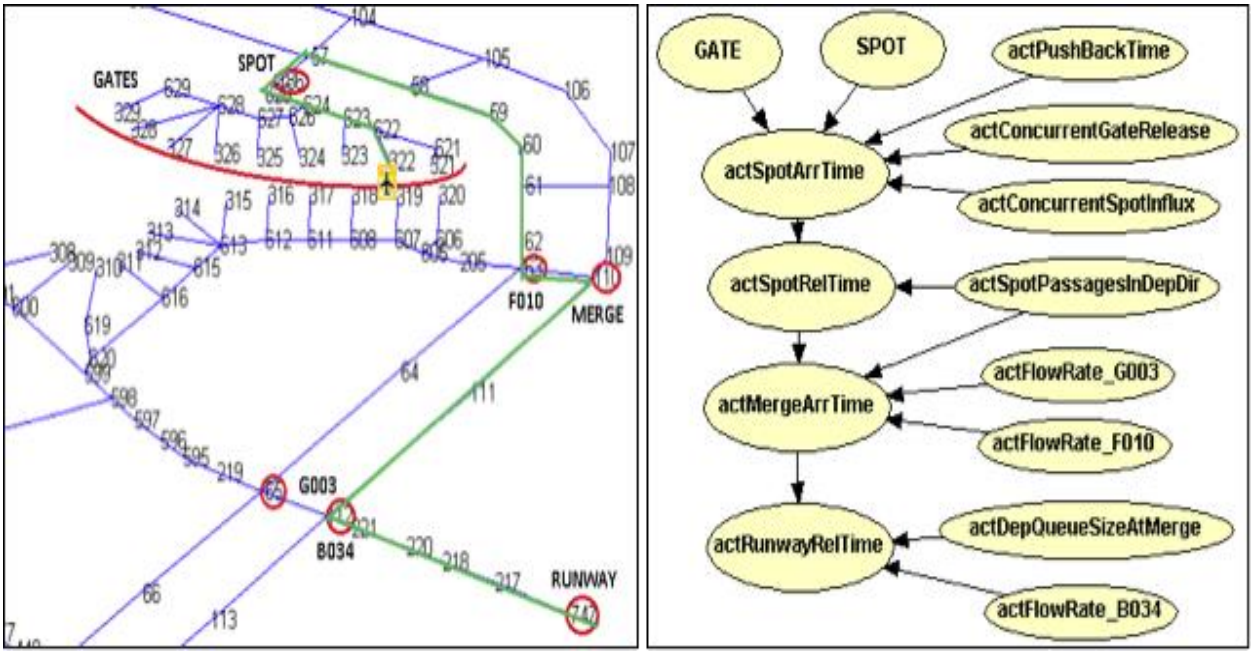


Figure 21. Phase I BN Structure from SME Consultation. Left, JFK Airport Snapshot with a Flight's Trajectory Highlighted in Green. Right, Phase I BN Structure.

As in Phase I, we (initially) learn a distinct BN model for each spot; that is, for departing flights that pass through a given spot. These are flights leaving from nearby gates which use the spot, which tend to have similar taxi paths to the runway. To learn the parameters for a particular spot-model, we therefore use the training data for all the flights that pass through that spot. We use the junction tree algorithm to create inference engines for each spot. The trained inference engine is then used to predict the futures for each flight in the test data by selecting the engine for the flight's scheduled spot.

In training each per-spot model, the spot random variable will always have the same value, and so, does not play an important role. In later work, we construct unified taxi time models for the whole airport, in which the spot random variable becomes significant.

5.1.2.4 Model Evaluation Metrics

The predicted and observed arrival times (for spot, merge and runway) are compared based on the following two metrics:

1. Prediction error: Root mean squared error (RMSE) of arrival time predictions with respect to actual arrival times.
2. Similarity between predicted and observed distributions: We use the p-values of Kolmogorov–Smirnov test (K–S test or KS test) as a measure of similarity. The higher the p-value is, the more similar the distributions are.

P-value is the minimum significance level (α) at which you reject the null hypothesis. The null hypothesis of the KS test is that the cumulative distribution function of the two compared samples comes from the same distribution. If α of the test is higher than the p-value, we reject the null hypothesis. Conversely, if α is lower, we accept the null hypothesis following the test implication. In this case, we conclude that the compared samples are similar in distribution.

During generation of multiple futures, a sample from the inferred distribution is predicted as arrival time. Hence computing RMSE by comparing samples to true arrival times does not complete the evaluation.

We also need to compare the distributions using a similarity metric. Thus RMSE and *p*-values together form a complete set of evaluation metrics.

5.1.2.5 Summary Algorithm

The pseudocode for the evaluation framework is shown in Figure 22. The airport consists of a list of flights $\Pi = (\pi_1, \pi_2, \ldots, \pi_n)$ and trajectories = $(\phi_1, \phi_2, \ldots, \phi_m)$. Any flight $\pi_i$in the airport will traverse through one such trajectory ϕ_j. Each trajectory consists of a set of important points in the 3D path, $\phi_j = (n_1, n_2, \ldots, n_k)$, denoted as nodes. All departing flights follow three important nodes: spot node, merge node and runway node. The list of spot nodes are denoted by Y. The simulation data $\hat{X} = (\boldsymbol{\Pi}, \boldsymbol{\Phi})$ consists of a list of flights and their trajectories. For each flight π_i, the data consist of a node name n_r and an arrival time t_r of the flight at the r^{th}node. In procedure SplitData, we group the flights based on the spot and create the Table of Factors as described in Section 5.1.2.1. We split the TOF data $\widehat{\boldsymbol{X}}$ into train $\widehat{\boldsymbol{Y}}$, and test data $\widehat{\boldsymbol{Z}}$ as described in Section 5.1.2.2. The train and test data are based on the spots: $\hat{Y} = (Y_1, Y_2, \ldots, Y_p)$ and $\hat{Z} = (Z_1, Z_2, \ldots, Z_p)$ where p is the number of spots. The training data Y_a for a spot a is used to create BN inference engines specific to that spot. The list of inference engines is stored in$\Theta = (\boldsymbol{\theta_1}, \theta_2, \ldots, \theta_p)$. The trained BN engines Θ are used for predicting the arrival times for each flight in a spot. The procedure Test retrieves the flight data (π) and trajectory data (ρ), and uses it as evidence in the trained engine (θ_{spot}) for the corresponding spot. Then, inference is run on the updated BN to predict the flight arrival times (PredictSamplingTimes). The predicted results ($\widehat{\boldsymbol{Z}}'$) are compared with the actual results ($\widehat{\boldsymbol{Z}}$) and performance metrics are computed as shown in the Evaluation procedure.

Algorithm 2.1: EVALUATION FRAMEWORK($\widehat{X}, \Gamma$)

procedure MAIN()
$\widehat{Z}, \widehat{Y} \leftarrow$ SPLITDATA($\widehat{X}$)
$BN \leftarrow$ CREATEBNS($\widehat{Y}$)
comment: BN structure is created
comment: We create a BN inference engine for each spot
for each $Y_{spot} \in \widehat{Y}$
do $\begin{cases} \theta_{spot} \leftarrow \text{TRAINBN}(Y_{spot}, BN) \\ \Theta \leftarrow \Theta \cup \theta_{spot} \end{cases}$
comment: We use the trained BN inference engine for each spot
for each $Z_{spot} \in \widehat{Z}$
do $\begin{cases} \theta_{spot} \leftarrow getEngine(\Theta) \\ \textbf{for each } \pi, \phi \leftarrow Z_{spot} \\ \quad \textbf{do} \begin{cases} Z' \leftarrow \text{TEST}(\pi, \phi, \theta_{spot}) \\ \widehat{Z'} \leftarrow \widehat{Z'} \cup Z' \end{cases} \\ eval_{results} \leftarrow \text{EVALUATION}(\widehat{Z}, \widehat{Z'}) \end{cases}$

procedure SPLITDATA($\widehat{X}$)
comment: Create a Table of Factors for each spot
for each $spot \in \Gamma$
do $\begin{cases} \textbf{for each } X \in \widehat{X} \\ \quad \textbf{do} \begin{cases} \Pi \leftarrow getFlights(X) \\ \Phi \leftarrow getTrajectories(X) \\ \textbf{for each } \pi \in \Pi \\ \quad \textbf{do} \begin{cases} \phi \leftarrow getTrajectory(\pi, \Phi) \\ \textbf{if } \phi.hasSpotNode(spot) \\ \quad \textbf{then } TOF_{spot} \leftarrow TOF_{spot} \cup \{\pi, \Phi\} \end{cases} \end{cases} \\ TOF \leftarrow TOF \cup TOF_{spot} \end{cases}$
comment: Split Table of Factors for each spot into train and test
for each $TOF_{spot} \in TOF$
do $\begin{cases} \textbf{comment: } \text{Randomly split TOF into train and test in 6:4 ratio} \\ TOF_{spot} \leftarrow random.shuffle(TOF_{spot}) \\ trainLength \leftarrow 0.6 * length(TOF_{spot}) \\ testLength \leftarrow 0.4 * length(TOF_{spot}) \\ Y_{spot} \leftarrow TOF_{spot}[1 : trainLength] \\ Z_{spot} \leftarrow TOF_{spot}[1 : testLength] \\ \widehat{Y} \leftarrow \widehat{Y} \cup Y_{spot} \\ \widehat{Z} \leftarrow \widehat{Z} \cup Z_{spot} \end{cases}$
return $(\widehat{Y}, \widehat{Z})$

procedure TRAINBN(Y_{spot}, BN)
$BN_{spot} \leftarrow$ LEARNPARAMS(Y_{spot}) **comment:** BN for each spot is learnt
$\theta_{spot} \leftarrow$ RUNINFERENCE(BN_{spot}) **comment:** Inference engine for each BN is created
return (θ_{spot})

procedure TEST(π, ϕ, θ_{spot})
$\theta_{spot} \leftarrow$ ENTEREVIDENCE(π, ϕ) **comment:** Update θ_{spot} with the evidence in π and ϕ
$Z' \leftarrow$ PREDICTSAMPLINGTIMES(θ_{spot}) **comment:** Spot, Merge and Runway times are predicted
return (Z')

procedure EVALUATION($\widehat{Z}, \widehat{Z'}$)
for each $Z, Z' \leftarrow \widehat{Z}, \widehat{Z'}$
do $\begin{cases} pvalues \leftarrow \text{RUNKOLMOGOROVSMIRNOV}(Z') \quad \textbf{comment: } \text{Kolmogorov Smirnov test is run for each flight} \\ error \leftarrow Z' - Z \quad \textbf{comment: } \text{Spot, MergeNode and Runway prediction error is calculated} \end{cases}$

Figure 22. Algorithm to Construct Table of Factors and Learn and Evaluate BN Model for Phase I BN Node Models.

5.1.3 Fixing the Bayes Net Toolbox (BNT)

During the initial experiments with Phase I approach, we studied the marginal distributions of transit time variables (spot, merge and runway arrival times). We observed a disparity between the marginals directly estimated from the data and the distributions of the samples generated from trained BN models. After many elaborate and thorough investigations we found a bug in the open source Bayes Net Toolbox (BNT) for MATLAB used for this work. We fixed the bug and the resulting toolbox produced correct results; that is, the predicted distributions matched the data distributions.

In the following subsection, we provide a brief description of the bug. We then compare the marginal distributions of some random variables in the table of factors data and trained BN model, in both original BNT and bug fixed BNT. We show that the marginal distribution generated by the bug fixed BNT matches the table of factors data while the marginal distribution generated by the original BNT shows a random behavior.

5.1.3.1 Summary of the BNT Bug

The following functions are buggy in the open source BNT toolbox found from https://github.com/bayesnet/bnt (available from Jan 2014).

- https://github.com/bayesnet/bnt/blob/master/KPMtools/find_equiv_posns.m
    - function p = find_equiv_posns(vsmall, vlarge)
    - The function assumes sorted order of vsmall and vlarge. In certain cases (NOT ALWAYS) this assumption is violated.
- https://github.com/bayesnet/bnt/blob/master/BNT/potentials/Tables/extend_domain_table.m
    - function B = extend_domain_table(A, smalldom, smallsz, bigdom, bigsz)
    - This function uses find_equiv_posns and thus suffers from similar issue: unsorted smalldom and smalldom.

5.1.3.2 Approach to Identifying the Bug

In order to highlight the behavior of the original and bug-fixed BNT, we compare the marginal distributions generated by the random variables like *spotArrivalTime*, *mergeNodeArrivalTime* and *runwayArrivalTime*. We generate the actual marginal distributions using the table of factors data. This is the true distribution. We assume that after parameter learning, the random variables in the trained BN model should also have a similar distribution.

5.1.3.3 Comparison Results for the Bug-fix

In Figure 23, we show the true marginal distribution of *spotArrivalTime*, *mergeNodeArrivalTime* and *runwayArrivalTime* variables generated from the table of factors data. We then generate the marginal distribution of the same variables from the trained BN model using the original BNT (shown in Figure 24) and the bug-fixed BNT (shown in Figure 25). Figure 24 did not match the true distribution. Figure 25, which shows the marginal distribution from the bug-fixed BNT, matches the true distribution and the P-distribution for runway arrival times (*actRwyArrTime*) as shown in Figure 20.

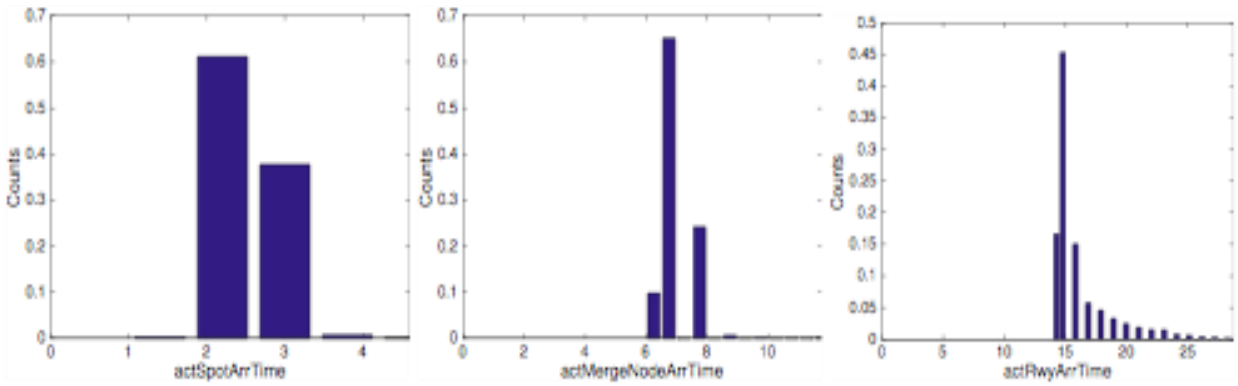


Figure 23. Histograms from Table of Factors Data

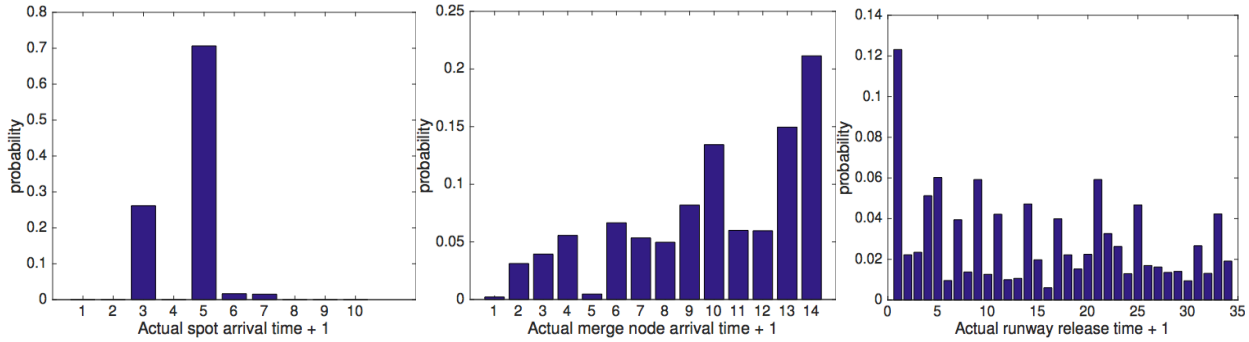


Figure 24. Marginal Distributions from Trained BN Models using the Original BNT

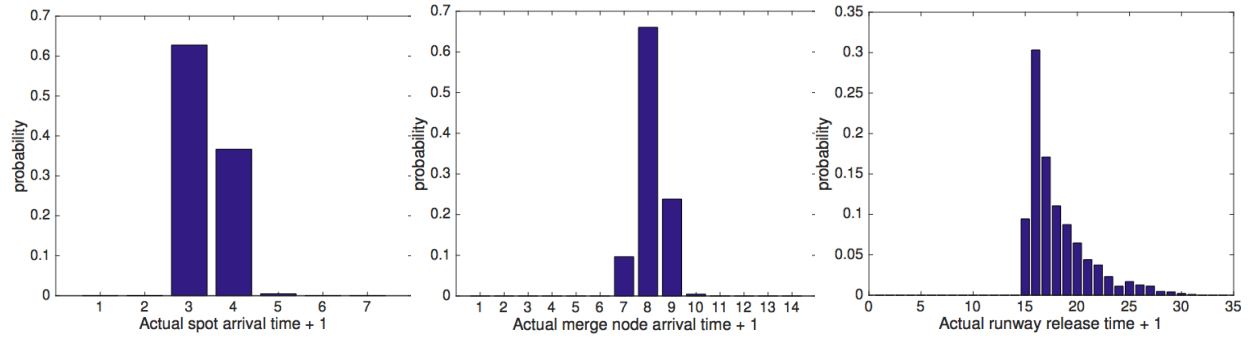


Figure 25. Marginal Distributions from Trained BN Models using the Fixed BNT

The right-most plot in Figure 23 shows some gaps in the histogram of runway arrival times from the data. This is an effect of the implementation of the histogram function in MATLAB for continuous data. We divided the times by 30 to match the distributions from the BN models (which come from discretized data). The histogram of resulting samples lead to some small empty bins appearing as gaps in Figure 23.

5.1.4 Sampling the Trained Bayesian Network Model to Generate Futures

A *future* is the collection of *predicted* 4D trajectories of multiple flights that is a plausible extension of the current state of the airport (at a scheduling cycle) into the future. A future contains the complete predicted 4D trajectory for each active or pending flight within the time horizon for traffic planning, beginning at the current time instant (e.g., SOSS simulation time step). PROCAST predicts the 4D trajectory of a departure flight by sampling the trained BN models of taxi time. The future is input to the PROCAST multi-airport traffic scheduling algorithm, which alters the 4D trajectories of the flights as needed by specifying the arrival fix crossing times, departure pushback times, and associated runway times of the

flights to comply with the capacity constraints of the runways and fixes. The output of the PROCAST scheduling algorithm is a *planned* 4D trajectory for each flight comprising the future. Different futures will have different predicted 4D trajectories for the same flights comprising the future, due to the random variations in the taxi times captured in the trained BN models. The PROCAST scheduling solution (i.e., the planned 4D trajectories) will also be different for different futures comprising the same set of flights.

The remainder of this section discusses the particular 4D trajectory times that are sampled for departures for the current PROCAST implementation, and alternative methods for sampling the BN models to predict the 4D trajectories of flights comprising the future.

5.1.4.1 Trajectory Times for Prediction

A future contains the complete predicted 4D trajectory for each departing flight that is currently pending, at the gate, or taxiing at the current instant in time (e.g., SOSS simulation time step). The 4D trajectory for each departure flight comprises its 3D path from gate to departure fix and times at the points along the path (e.g., gate, spot, runway, fix). Depending on the current state of the flight (pending, at the gate, or taxiing), some or all of the times at the gate, spot, runway and fix will be predicted using the BN model.

The current implementation of arrival-departure scheduling in PROCAST for this project, only the ETA of the departure to its takeoff runway (*runway time* in the following discussion), and the gate pushback time (*readiness time* in the following discussion) are required for scheduling.

For departures that are pending, or at the gate, the readiness time is a time, in the future, when the aircraft will be ready to depart from the gate. The readiness time of each departure is given as input data to the SOSS simulation. The runway time can be computed as the sum of the readiness time and the predicted gate-to-runway taxi time. For taxiing departures, the runway time is the predicted time of arrival at the runway, given the actual gate pushback time, and the progress reflected in the current state.

PROCAST scheduling relies on predictions of the gate-to-runway taxi times of both taxiing and non-taxiing departure flights in order to schedule the runway times, and, for departures at the gate and pending, the gate pushback times. For PROCAST, we first predict the *distributions* of these transit times, then sample the distributions repeatedly, to construct multiple futures based on the current airport state.

The 4D trajectory of each departure contains five transit time variables, *actPushbackTime*, *actSpotArrTime*, *actSpotRelTime*, *actRwyRelTime*, and *actMergeNodeArrTime* and the associated the states of all variables in the Phase I BN (Figure 21). Depending on the current location of the aircraft, one or more transit time fields in its 4D trajectory may not yet have a value, or all transit time fields may have values. Departures which are ready to pushback at the gate or taxiing on the airport surface towards the runway will have some values missing, thus require runway arrival time predictions. Departures which have already departed from the runway and are approaching departure fixes will have all the fields filled in for their trajectory.

We use the *perturb_trajectory* method to fill in one or more missing fields among the five transit time variables for departures. We predict these variables using a trained BN model (as shown in Figure 21). Since the BN models the transit times between the gate, spot, merge, and runway, the value of *actPushbackTime* is added to the predicted duration from BN to estimate actual arrival times at the spot, merge, or runway.

Generating futures is a two-step approach. First, we supply all non-transit time variables to the trained BN model as evidences and use the junction tree inference algorithm [KD09] to get the posterior distribution of an enquired transit time variable. Then we generate a sample from this distribution to obtain a transit time value and include it with the appropriate node in the 3D path to generate the 4D trajectory.

Sampling is done in two different ways: stage-wise sampling and posterior sampling. Each is described next.

5.1.4.2 Stage-wise Sampling

In this scheme, inference and sampling are repeated sequentially to predict all three transit-time variables. First, the spot node arrival time is sampled from the distribution. Next, the sampled value of spot arrival time is provided as evidence to infer the merge node arrival time. Finally, the same process is repeated to sample the runway node arrival time, using the sampled arrival times for the spot and merge nodes as evidences. To sample the three variables (spot, merge and runway), three different inference engines are created.

5.1.4.3 Posterior Sampling

In this scheme, we create one junction tree inference engine using only non-transit time variables as evidences. We sample the posterior distributions of spot, merge and runway arrival times independently to fill in the missing fields in the trajectory. As some parents of runway (spot and merge) can be absent from the evidence set, the inference algorithm marginalizes over all the missing parents to estimate the posterior distribution.

5.1.4.4 Comparison

The posterior sampling method is much faster than the stage-wise method because constructing a junction tree inference engine is computationally intensive. Perturbing a trajectory requires predicting values for one or more (1, 2 or 3) transit time variables (*actSpotArrTime*, *actMergeNodeArrTime*, *actRwyRelTime*). If all three variables need to be predicted, the stage-wise sampling method requires constructing two extra junction tree inference engines. An empirical comparison of these two methods is presented later (Figure 28 and Figure 29).

5.1.5 Addressing Prediction Conditions Having Zero Probability Values

When we encounter conditions for prediction not captured in the training data, the trained BN model will represent this condition as having zero probability, thus may not be able to make a prediction for that condition. During prediction from a trained BN model, if a new combination of evidence variables is provided, the inference algorithm may output zero distribution (zero probabilities for all states) if care is not taken. This is a fundamental problem, especially with discrete data, as in our case. We attempt to alleviate this problem by a few approaches, as discussed below, including supplying a default value, using a prior distribution, increasing the bin size for discretization, and using fewer evidence variables for prediction.

5.1.5.1 Supply a Default Value (Phase I Approach)

In the Phase I approach, whenever zero probabilities were encountered during prediction, a default (predefined) value was output. The default value was tuned experimentally. For a fixed time bin size and Phase I dataset, this approach performed reasonably well. However, later experiments show the drawback of this method.

5.1.5.2 Bayesian Network with a Prior Distribution

The BNT provides an option to specify the Dirichlet prior with different weights. We experimented with different weights between 0.05 and 2 for JFK spot node VA_001. The results are shown in Figure 26.

The baseline RMSE with no prior for stage-wise and posterior sampling is shown as a straight line for comparison. In the “no prior” method, we use a fixed value based on the domain knowledge. The sampling was done using both the stage-wise and posterior methods. During prediction, as expected, we found that there were no zero Probability Mass Functions (PMFs). It can be seen that for varying weights, the errors show a random behavior for spot, merge and runway. The weight value of 0.7 showed the lowest RMSE, 68.7, using stage wise sampling for runway which is lower than the baseline RMSE, 88.62, using stage-wise sampling and 72.63 using posterior sampling when no prior was used.

It might be that the BN with Dirichlet prior would give better predictions if it was trained with a bigger data set. Figure 26 shows the effect of varying training sample sizes for runway when the Dirichlet weight is 0.7. We can see that the RMSE increases for smaller sample sizes. The RMSE becomes equal to the RMSE of the "no prior" method as we increase the training sample size. This behavior can be seen in both stage-wise and posterior sampling methods.

In conclusion, we found that the transit time prediction results using a prior are not encouraging; not convincingly better than the baseline with no prior.

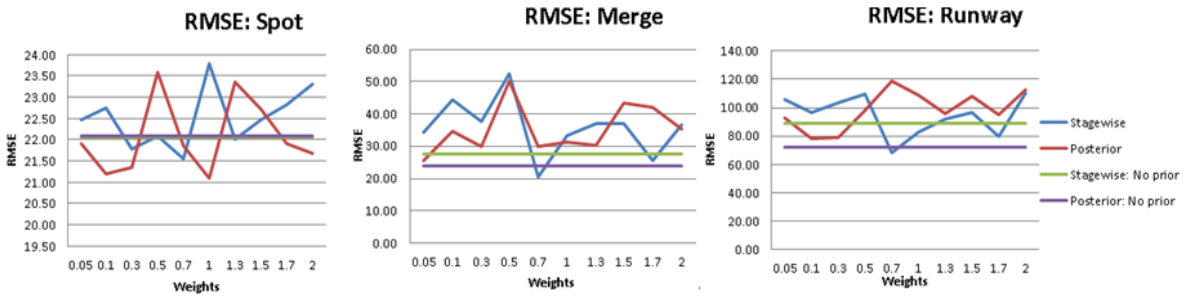


Figure 26. RMSE for Varying Dirichlet weights. The Straight Line is the RMSE When No Prior Is Used.

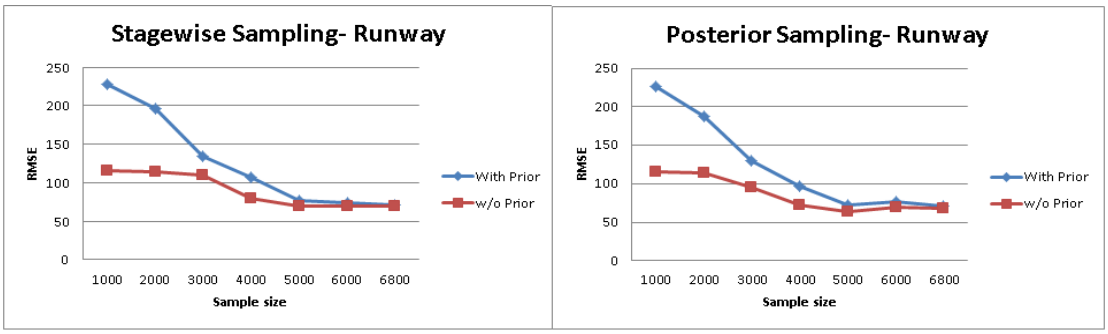


Figure 27. RMSE Versus Sample Size With Dirichlet Weight of 0.7. RMSE of Dirichlet Prior Method is Large For Smaller Sample Sizes, Decreases To "No Prior" Method With Increasing Sample Size.

5.1.5.3 Larger Bin Size to Address Sparseness of Training Data

The bin size for discretizing the transit times determines the number of states of the associated random variables in the BN. Smaller bin size, despite creating lower discretization error, increases the number of states of the random variables. This increases the sizes of the CPTs. Hence for a fixed amount of training data, smaller bin sizes will result in sparser CPTs. During prediction, sparse CPTs will produce posterior probability distributions that are zero.

We compared the effect of different bin sizes from 10 to 50 seconds in terms of prediction error. Table 2 shows the RMSE of three transit time variables' prediction for different time bin sizes. We present the results for both sampling methods.

Table 2. Root Mean Squared Error of Three Transit Time Variables' Prediction for Different Time Bin Sizes

	RMSE of Stage-wise sampling			RMSE of Posterior-based sampling		
Time bin size (sec)	spot	merge	runway	spot	merge	runway
10.0	9.01	23.94	99.52	10.83	19.55	83.96
20.0	14.47	19.15	79.13	15.19	19.21	73.79
30.0	20.71	23.05	71.88	20.49	23.67	67.74
40.0	29.01	32.54	65.27	28.60	32.37	65.34
50.0	37.16	41.64	72.41	36.32	39.67	70.34

Figure 28 presents the RMSE of predicted runway transit times as a function of time bin size. The results indicate that the RMSE of predicted runway transit times using a stage-wise sampling method varies significantly with time bin size. In contrast, the posterior method shows smaller variations. For both sampling methods, the trend with increasing time bin sizes is that the prediction error decreases, with the minimum error obtained using time bins of 40 seconds. Beyond that, larger time bins yield increased errors. Between the two methods, the posterior-based method performs better than the stage-wise method.

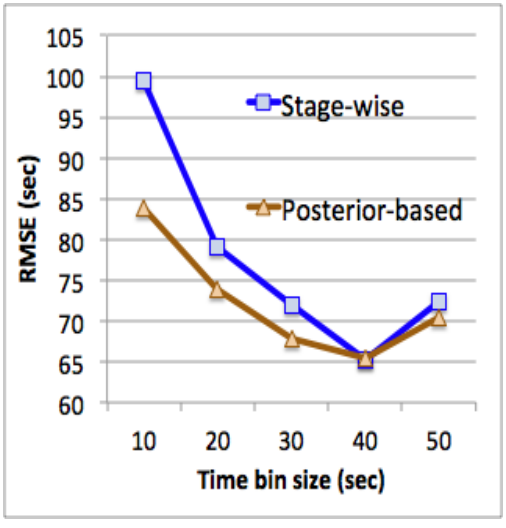


Figure 28. Effect of Time Bin Sizes on Prediction Performance

It is bit unclear why 30 to 40 second time bin sizes were well-performing. However, the trend seen here is consistent with the hypothesis that zero posterior probability distributions (and the naive correction to them) are a major source of prediction error in the Phase I model, with discretization error becoming a significant factor only at the largest time bin sizes. In the Phase I approach, the pre-defined prediction values for zero probability instances were tuned for 30 second time bin size. Thus, for smaller or larger bins, prediction error increases.

5.1.5.4 Prediction Using Less Evidence (Fewer Inputs)

To eliminate the problem of zero probability instances during inference from the trained BN engine, we implemented a scheme to methodically reduce the evidences such that we get non-zero probabilities from the inference engine. Recall that zero probabilities arise when the combination of values supplied as evidence do not occur in the training data. If we ignore some of those variables, then we are more likely to have at least some training data that matches the remaining ones. Thus, with fewer evidence variables, it is more likely to get a non-zero posterior probability distribution during inference.

We sort all the evidence variables in decreasing order of the number of states. We start the inference process with all evidence and estimate the posterior distribution for the enquired variables (spot, merge and runway). If a zero distribution is returned, we incrementally reduce the evidences in the aforementioned order until a non-zero distribution is returned.

We compared this scheme with the default value approach for the posterior sampling case (since this method performed better in previous experiments, see Figure 28). We used simulation data of flights through one spot. Figure 29 shows the comparison results with different time bin sizes.

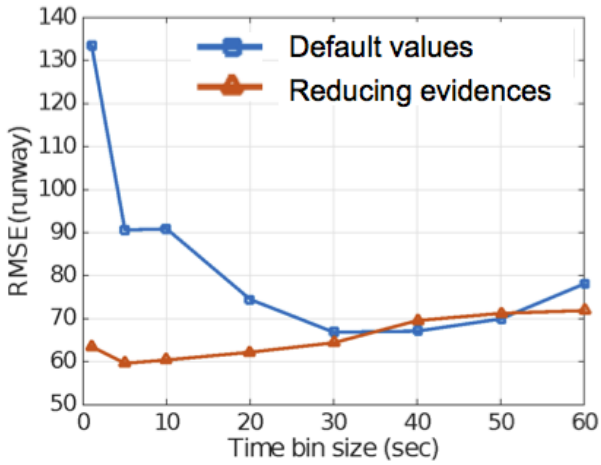


Figure 29. Comparison of Phase I Approach and Our Scheme of Reducing Evidences for Solving Zero Probabilities Problem (with Posterior Sampling)

We observe that the prediction error remains low with this scheme for a wide range of time bin sizes. Moreover, best prediction performance is achieved with a time bin size of 5 seconds. With decreasing bin sizes, the discretization error decreases. And with bins smaller than 5 second, data becomes too sparse to populate the conditional probability tables of the BN. Thus RMSE increases for the 1 second time bin size.

Overall, the scheme with reducing evidences shows superiority to the method of using default values in most cases. The minimum RMSE is about 60 seconds (for 5 second time bins). Therefore we adopt this method for all later experiments with discrete data.

5.1.6 Using Continuous Variables in the Bayesian Network

Representing transit time variables as continuous random variables improves modelling accuracy, and continuous variable models cannot produce posterior distributions that are zero. However, using the available packages (BNT and *bnlearn*), a switch to continuous variables adds complexity to PROCAST in the training of BN models and interfacing with the arrival-departure scheduler.

We investigate the continuous representation of transit times in the BN. The BNT toolbox implements the training and inference for continuous variables by separate methods from discrete variables. These methods are likely to be affected by a bug similar to the one described in Section 5.1.3. Hence we avoid BNT for this experiment and use the R *bnlearn* (Scutari 2010 [MS10]) package instead.

The BN structure remains the same as in Figure 21b. We learn the model parameters using the *bn.fit* method and predict the runway arrival times using the *predict* method. This method needs the values of all parent nodes to predict any BN node. Therefore we first make predictions for spot node, then merge node and finally runway node which can have both spot and merge nodes as its parents. If a flight has passed a spot and/or merge node, its arrival time to the spot and/or merge node are known. In this case we use the actual arrival times of spot or merge nodes to predict runway arrival time.

Using the same training and test sets (which are generated from Phase I dataset for one spot) as in the previous experiment (Figure 30), the prediction errors are a Mean Absolute Error (MAE) of 44.5 seconds and RMSE of 34.7 seconds. The results are much improved compared to the previous experiment using discrete data. Can using continuous random variables make such a big difference in prediction? We investigate this question next.

Apart from using continuous versus discrete variables, there are two major differences in the last two experiments: MATLAB BNT package versus R *bnlearn* package, and sample versus predict. To compare the two BN packages, one would need to fix the possible bugs with continuous variables in BNT. Due to the limited scope of this project, we leave that as future work.

However, we compare the *sampling* and *predicting* schemes in BNT with discrete variables. In the *sampling* case, the prediction is a sampled value from the posterior distribution and in the *predicting* case the posterior mean is the predicted value. The following example comparison illustrates the effect of sampling versus predicting:

> *Sample*: RMSE (spot, merge, runway) = 20.584028, 23.754675, 63.131729
>
> *Predict*: RMSE (spot, merge, runway) = 17.642107, 18.811627, 47.712309

We observe a significant drop in prediction error for runway arrival in the *predicting* case. Since the objective of model training is minimizing a squared error based cost function, it is expected that the *predicting* scheme would provide more accurate predictions. The main purpose of *sampling* is to generate multiple futures as described in Section 5.1.4.

The previous experiment demonstrates the additional error introduced by sampling. We need to do a fair comparison between continuous and discrete datasets. We trained two models using continuous and discrete data and evaluated them under the *predicting* scheme. Figure 30 and Figure 31 show the mean and standard deviations of the prediction errors for the *predicting* scheme with continuous (*bnlearn*) and discrete (BNT) cases respectively.

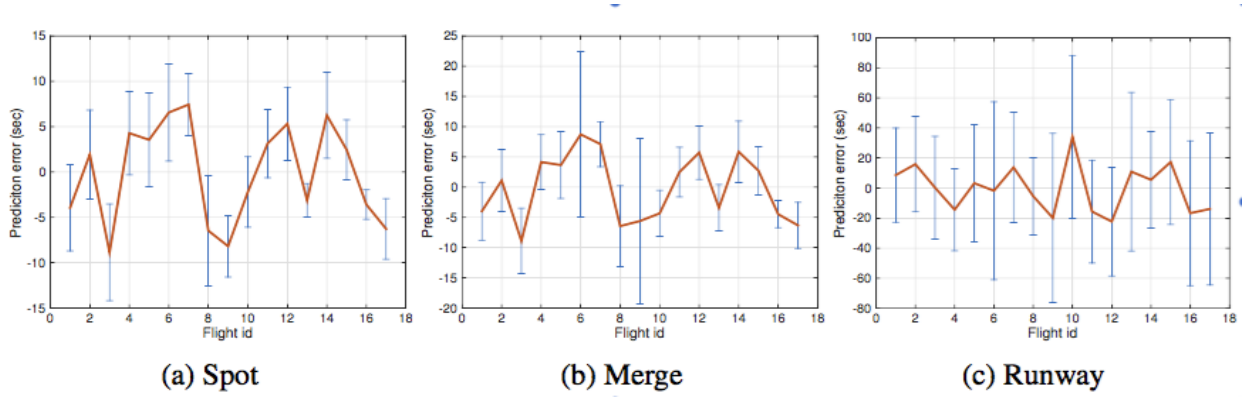


Figure 30. Prediction Errors Using Continuous Random Variables and R *bnlearn* Package (Spot = DA-015). Error Bars Represent the Variation over Multiple Simulations for this Spot.

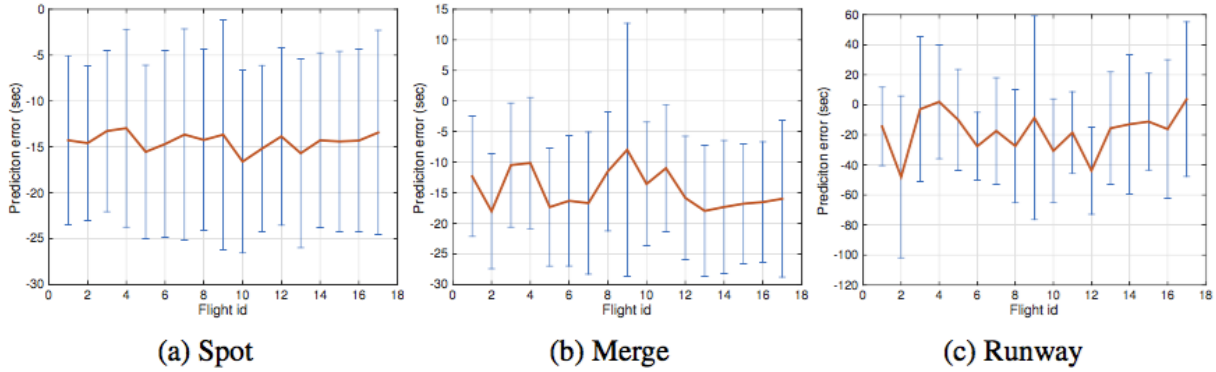


Figure 31. Errors in Predictions using Discrete Random Variables and Posterior Mean Method with MATLAB BNT (Spot = DA-015). Error Bars Represent the Variation over Multiple Simulations for this Spot.

The mean predictions in Figure 30c are around zero. This is not the case in Figure 31c where a bias in prediction is seen. Since we observe superior performance using continuous variables and *bnlearn* package, we use this method for predictions using Phase II data.

5.2 Generalized Bayesian Network Taxi Time Models for Arbitrary Airports

This section presents our Phase II methods and results of creating a generalized approach to Bayesian network modeling of taxi times for an arbitrary airport and applying the method to the JFK, EWR and LGA airports in the New York metroplex. Our work addresses a number of critical challenges and makes a number of fundamental contributions, including generalization to arbitrary airports, feature engineering, handling large and sparse data, and creating and using a complex BN model, summarized below:

- **Generalization to arbitrary airports:** The Phase I BN is specifically designed for the JFK airport. In Phase I design, subject matter experts played a key role in determining the nodes for the BN. It is done by picking the "important junctions" that contribute to the flight delay. Our main goal in Phase 2 is to generalize the BN to support any new airport. In order to do this, we did not want to rely on the subject matter expert's choice of nodes. So we follow a data driven approach to choose the "important junctions" in the airport that contribute to the traffic.
- **Feature engineering:** We had to devise a new approach to collect the traffic count (the number of flights passing through a route segment during a time period). Then, we pick the segments with

more number of flights as heavy traffic route segments. These chosen segments are used as the "important junctions" that contribute to the delay.

- **Large and sparse data:** Due to the above feature engineering, we end up with large number of features. Some of the features may not have traffic counts for that training data. This can cause the conditional probability tables to be large and sparse.
- **Complex BN model:** Instead of relying on subject matter experts on creating the BN structure, we use the structure learning algorithms to create BN structures from the given set of features. The resulting BN model can be complex where a node can have any number of parents and children making it difficult to comprehend the BN.

The multiple-airport approach presented in this section is general and works for any airport. By using a data-driven machine learning approach, we have developed a general transit time method that relies on Bayesian network structure learning. The method is experimentally demonstrated using data from three airports – JFK, EWR, and LGA. For JFK, we compare the Bayesian network structure learning method with the Bayesian network parameter estimation method used in Phase I. Experimentally, we find improved precision and accuracy when using structure-learned Bayesian network models.

This section begins with a description of the training data sets for Phase II (Section 5.2.1). We then present extension of the BN modeling and prediction framework to arbitrary airports, first by summarizing the features of the Phase I BN models (Section 5.2.2), then generalizing these features (Section 5.2.3). We present enhancements to the evaluation framework for creating and evaluating the the BN models (Section 5.2.4). We present learning the BN model structure from data (Section 5.2.5) and using the models for prediction (Section 5.2.6). We present experiments evaluating the prediction accuracy and sensitivity of the Phase II models (Section 5.2.7). We discuss the implementation of the models in PROCAST (Section 5.2.8).

5.2.1 Raw Phase II Data

In Phase II, new airport simulation data was generated using the enhanced SOSS. Although the simulation files had similar information to the Phase I simulation files, there were a few major differences. First, separate simulations were performed using traffic scenarios based on three actual days in 2012: May 13, June 11 and September 5.

Before each run of the simulation, the traffic scenario for that day is perturbed by adding a distinct Gaussian random value to the original start time for each flight: the gate pushback time for departures or the arrival fix crossing time for arrivals. A separate Gaussian distribution is used for each of the three airports, for each of the three months, and for arrivals and departures, yielding 18 distinct distributions. The standard deviations of the distributions are derived from Out-Off-On-In (OOOI) data sets from the FAA, for each of these airports, for each of these months, but from the year 2009. For example, we extracted from the OOOI data for all departures from EWR in May 2009, and computed the standard deviation of the delays from the data. That standard deviation is then used to perturb departure start times in the EWR May scenario.

Processing all the simulation files from the Phase II simulations of the New York metroplex was more of a computational challenge than in Phase I because the amount of simulation data was much higher:

- Each Phase II simulation is 24 hours long, while a Phase I simulation is 6 hours long
- 1000 simulation runs were conducted for each combination of day and airport, for a total of more than 9000 simulation output files. In contrast there are only 501 SOSS output files in Phase I.
- The formatting of Phase I and II simulations are slightly different. Every flight's trajectory had a few airborne nodes added. Hence, we had to pre-process the files accordingly to comply with the existing Phase I parser scripts.

Lastly, the simulations from the different traffic days have different sets of flights. These sets are not necessarily mutually-exclusive. Traffic conditions in the airport in different dates can be very dissimilar, and using one day as training and another for test data can result in poor prediction. Therefore, care must be taken in developing and evaluating the BN models of taxi time with this data.

5.2.2 Features of the Phase I JFK Model

Features are the random variables pertaining to a given departing flight. These features are the BN model nodes that appear in a Bayesian network. Figure 32 presents the features, categorized as temporal and non-temporal, for the BN models of JFK taxi times developed in Phase I.

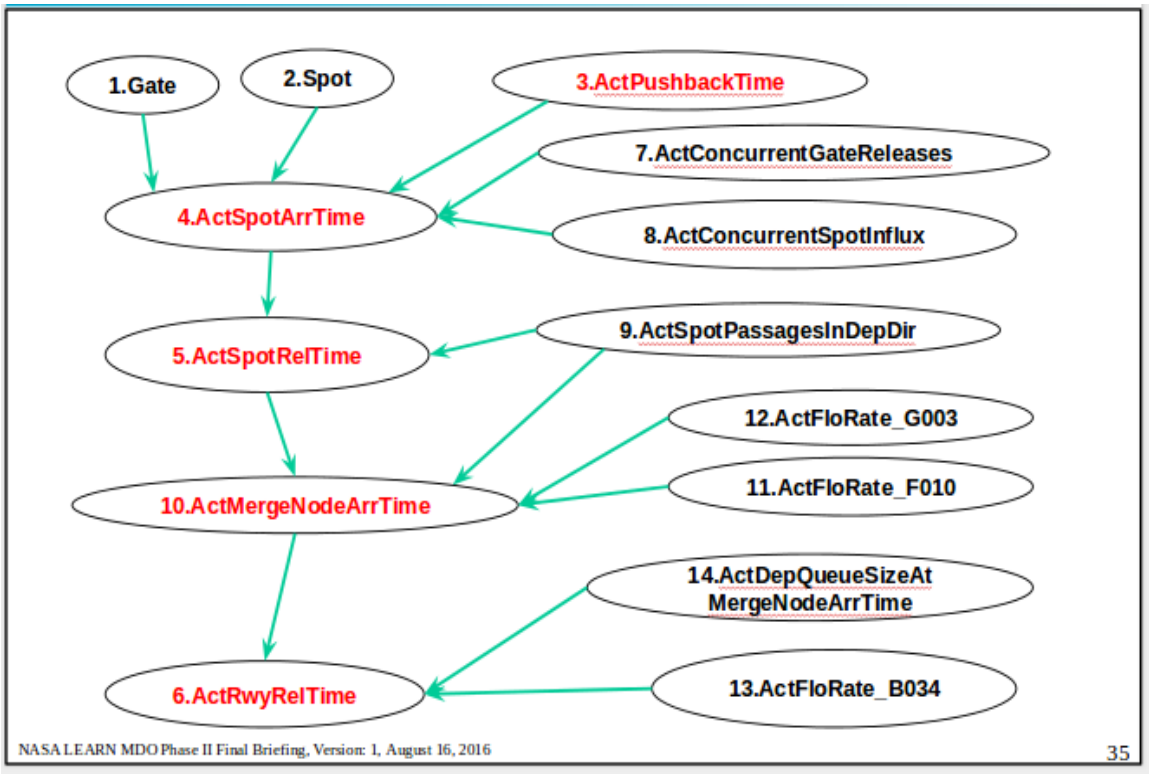


Figure 32. Bayesian Network Nodes and Structure for JFK Developed in Phase I. Red-colored Nodes are the Temporal Features and Black are Non-temporal Features.

Some of the Phase I features were specific to the JFK airport and its node-link model developed for SOSS (e.g. *ActFlowRate_B034*). Designing a set of such features requires a subject matter expert's advice, as well as trial-and-error. It is desirable to minimize dependency on such hand-crafted features, so that we can easily extend the model to other airports. So, in Phase II, we analyze these hand-crafted features and replace them with generalized features so that we can easily extend the models to other airports.

Features generated for JFK during Phase I can be seen in the BN structure shown in Figure 32. The features are grouped and briefly explained below.

5.2.2.1 Gate and Spot Features

Random variable G*ate* indicates the gate from which the given aircraft departed. The random variable *Spot* indicates the spot node in the aircraft's 3D path.

5.2.2.2 Temporal Features

Temporal features are the random variables that capture the aircraft's transit times. These transit time variables are calculated relative to pushback time of the aircraft (the *ActPushbackTime* feature in the BN model). These are:

- *ActSpotarrivalTime* and *ActSpotReleaseTime*, which measure the time taken to arrive at, as well as depart from the spot node, from the moment of pushback from the gate;
- *ActMergeNodearrivalTime*, which is the transit time between merge node arrival and pushback time; and finally,
- *ActRunwayReleaseTime*, which is a transit time between the pushback and the departure from the runway node. Essentially, these capture the timeline of an aircraft's trajectory.

5.2.2.3 Ramp Area Traffic Features

This group comprises three random variables which measure the traffic near the departure gate and the adjacent ramp area:

- *ActConcurrentGateReleases*, which is a count of the flights released from the same gate area in a small time-window surrounding the gate release time, and,
- *ActConcurrentSpotInflux*, which counts number of arrival flights crossing the spot into the same gate-ramp area in a time window surrounding gate release time.

The above two features capture factors influencing the aircraft's spot arrival time, once it has pushed back from the gate.

A third variable, *ActSpotPassagesInDepDir*, measures the number of aircraft passing *in the taxiway*, adjacent to a particular spot (e.g., node *DA_015* in the node-link model of JFK for SOSS). When a departure exits this spot and reaches the taxiway, it will turn, always in the same direction. The traffic count for this variable is restricted to flights travelling along the taxiway (but not exiting the spot) in the direction that a departure from the spot would turn.

5.2.2.4 Queue Traffic Features

Once the aircraft cross the merge node, they form queues at nodes prior to the runway node before they are allowed to depart from the runway. The queue traffic feature measures the traffic conditions in this segment between merge node and the runway node. One such queue traffic feature is the random variable *ActDepQueueSizeAtMergeNodeArrTime*, which counts the number of flights that have crossed the merge node earlier than the given aircraft but not yet taken off from the runway at the time window surrounding the merge node arrival. The number of flights in the departure queue affects the runway departure time of the aircraft currently at the merge node and that is what is being captured by this feature.

5.2.2.5 Taxiway Flow Rate Features

The taxiway flow rate features *ActFloRateG003, ActFloRateF010* and *ActFloRateB034* capture the traffic across certain segments in the taxiway between the spot node and merge node in the SOSS node-link model of JFK, in the time window surrounding spot node arrival time. The traffic across segments in taxiway affects the flight's merge node arrival time, since these segments lie in between spot node and merge node. Specifically, these measure the traffic across segments formed by the intermediate nodes, namely, *G003*, *F010* and *B034*. These intermediate nodes were selected by subject matter experts familiar with the JFK airport.

5.2.3 Generating a Generalized Set of Features

To extend the model to other airports, we seek to replace these hand-crafted taxiway flow rate features with generalized features that do not depend on the inputs of subject matter experts. So, we analyzed

these taxi flow rate features to understand their contribution to the BN models of transit time and to explore what they can be replaced with. In essence, these features measure the traffic at some point on the taxiway, in some window of time (relative to the given flight's timeline). For example, the features *ActFloRateG003* and *ActFloRateF010* capture the traffic across certain three-node segments containing intermediate nodes *G003* and *F010* during the time window of spot arrival for the given flight. Also, we observed that these nodes stand out as having a large amount of total traffic.

We leverage this idea to replace these hand-crafted features with a set of generalized features that measure traffic across high-traffic taxiway links in the airport link-node model for SOSS. From the simulation data, we compute traffic (arrivals and departures) across all the node-pairs (links) in the SOSS airport link-node model. Figure 33 shows the traffic distribution across the links. Negative link ids represent the traffic in the opposite direction for the same link-id.

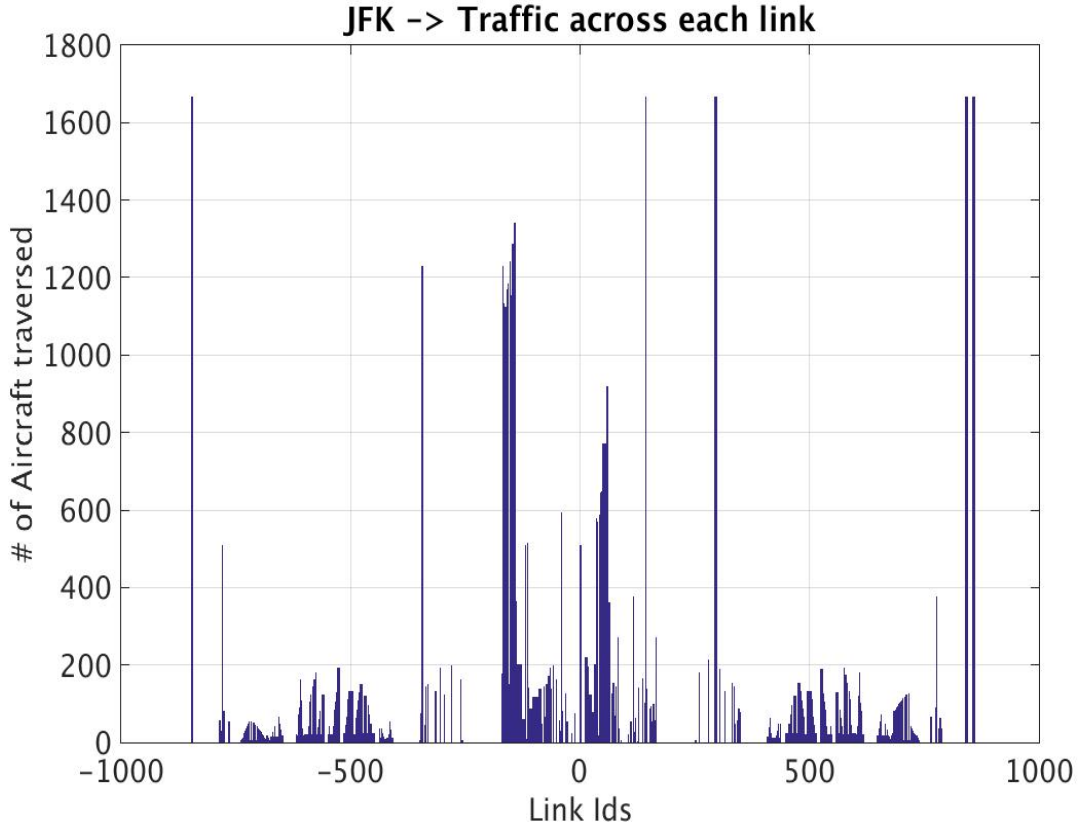


Figure 33. Traffic Across Links for JFK - Phase II.

We see per-link total traffic levels differing by more than two orders of magnitude. We consider links with traffic above a certain threshold for the feature generation, and call those links *high-traffic links*. We develop the following procedure to create a set of features based on traffic across *high-traffic links*:

- We compute traffic statistics for each link in the airport link-node model. Links are directional, so each airport link results in two features. Experimental results showed that top 50 links provided good prediction performance, so we pick 50 of the links with the highest traffic counts as *high-traffic links*. In addition, if structure learning is used to determine the structure among the variables in the BN model, using more features makes BN structure learning much slower due to linearly increasing computational complexity with increasing number of BN variables [DR11].

- We consider key moments in a flight's timeline, e.g., gate release time, spot arrival time and merge node arrival time, and we use a time window surrounding these key moments to compute traffic across all high-traffic links in the airport.
- Traffic across each of the high-traffic links is calculated, specifically in the time windows of spot node arrival and merge node arrival. So we end up with 100 features: 50 features for spot node arrival and 50 features for merge node arrival time window.

The set of these traffic features captures behavior similar to the hand-crafted features, specifically, *ActFloRateG003*, *ActFloRateF010* and *ActFloRateB034*, as well as the similar *ActSpotPassagesInDepDir*. So we replace the features by these newly computed 100 features. However, the factors that influence the spot arrival from gate release are the traffic in the same gate area and the traffic influx into the same gate area through the spot. This is effectively captured through *ActConcurrentGateReleases* and *ActConcurrentSpotInflux*. So we retain these features.

Table 3 shows the table of factors data structure for this Phase II model. Compared to the TOF data structure for the Phase I model listed in Table 1, two new fields are introduced: *Traffic_spot_arrival* and *Traffic_merge_arrival*. These fields contain the map structure where the keys are the link-ids and values are the traffic count across that particular link during the corresponding time window.

Table 3. Table of Factors Data Structure – Phase II.

	1	2	3	4
CallSign	JBU733	JBU1729	JBU733	JBU1729
Gate	Gate_05_016	Gate_05_021	Gate_05_016	Gate_05_021
Spot	DA_015	DA_015	DA_015	DA_015
....				
ActSpotArrivalTime	776.0200	3089.9800	1568.0200	3576.9800
DepQueueSizeAtMergeNOde	0	2	2	2
Traffic_spot_arrival	50*1 Map	50*1 Map	50*1 Map	50*1 Map
Traffic_merge_arrival	50*1 Map	50*1 Map	50*1 Map	50*1 Map

Figure 32 shows the Phase II BN with hand-crafted features. Figure 34 highlights the BN model nodes that are being replaced or removed from the Phase II BN and Figure 35 shows the new BN for Phase II which is a combination of the old features and the new generalized set of features.

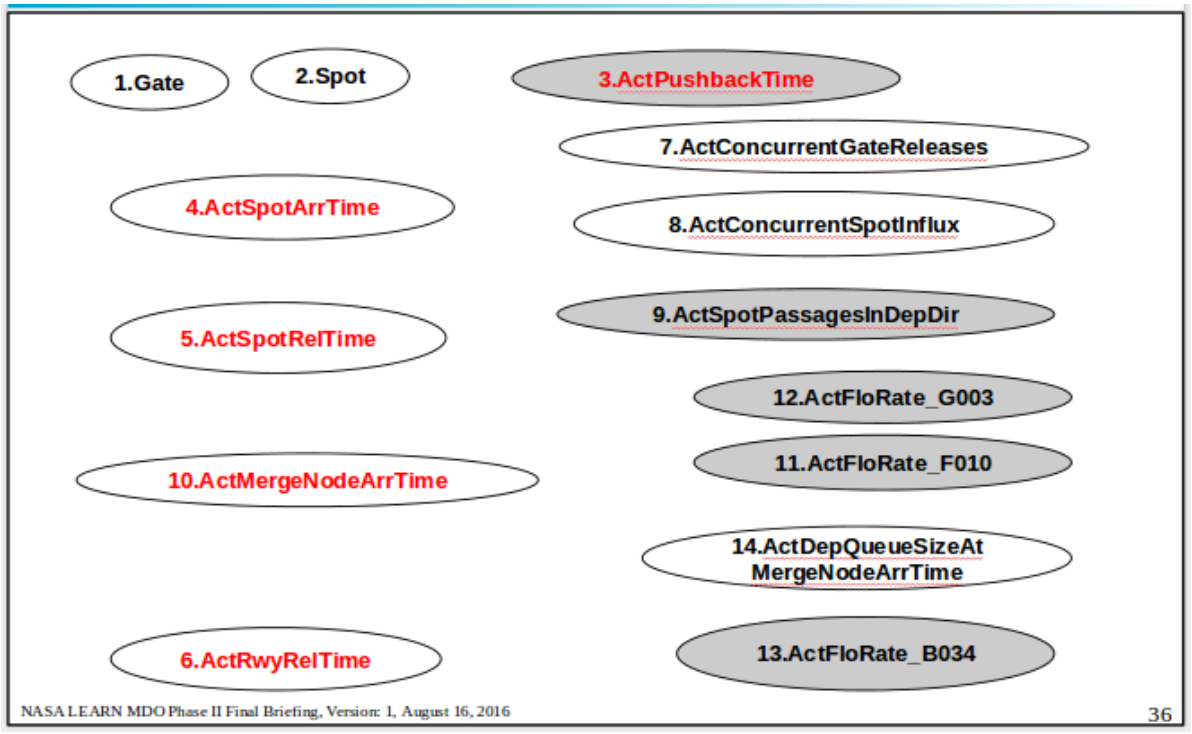


Figure 34. Features of Phase I BN Model to be Replaced or Removed in Phase II BN Model are shown in Grey.

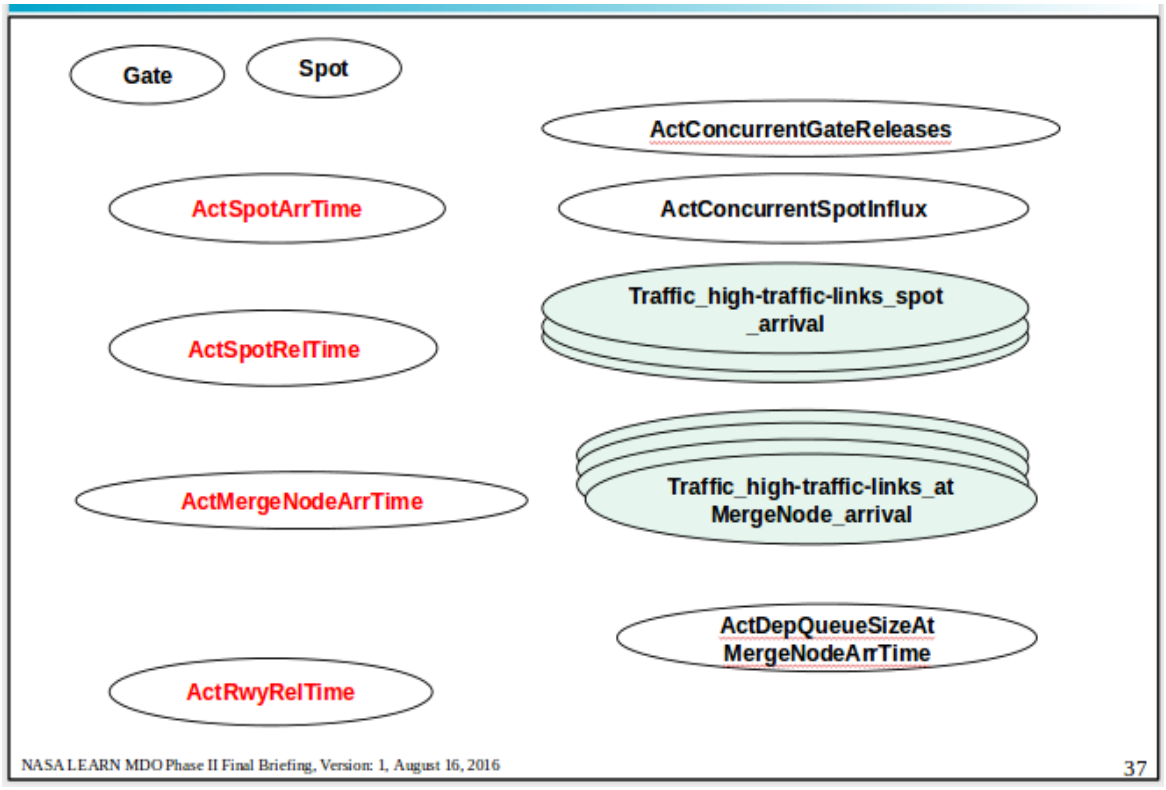


Figure 35. Features added in Phase II. Highlighted in Pale Blue are the Generalized Features Replacing the Hand-crafted Features of Phase I.

The result is a generic BN model structure which can be applied to modeling any arbitrary airport. This generic model structure can in turn be applied to generating BN models of taxi times for LGA and EWR airports in the New York metroplex.

5.2.4 Enhancing the Evaluation Framework

For Phase II of the project, we want to create three different BN models of the taxi times, one for each airport (JFK, EWR and LGA). We accomplish this by enhancing the evaluation framework for BN modeling that we developed for the Phase I BN models for JFK, as depicted in Figure 36.

The two main challenges in generating the models and processing Phase II data are 1) the applicability of the process to all three airports and, 2) processing SOSS simulation output files generated by simulating different traffic days, and multiple simulations for each traffic day, for each of the three airports. For each airport, the SOSS simulation data have different sets of flights among the three different traffic days in May, June, and September of 2012. For compact representation, we suppress the dates in Figure 36.

We solve the first challenge by inserting BN structure learning in the model training step. Without a pre-defined structure, this framework can generalize to any airport's data. Learning the parameters of the conditional probability tables remains as the next step of structure learning.

We address the second challenge by grouping SOSS simulations from multiple days. Equal numbers of simulation results for each traffic day are grouped for every airport. Different airports' data are processed separately to train three different models, one for each airport. The grouped SOSS simulation data are converted to a table of factors which involves feature extraction as mentioned in Section 5.2.3. These features consist of traffic along all links in a given airport. Learning a BN model from such a high number of variables is very computationally intensive. Hence we perform feature selection before structure learning.

Figure 36 shows all the processing steps of the enhanced evaluation framework: grouping SOSS simulations, extracting features, selecting important features, creating train and test sets, leaning BN models, predicting for test set, and evaluating performance metrics.

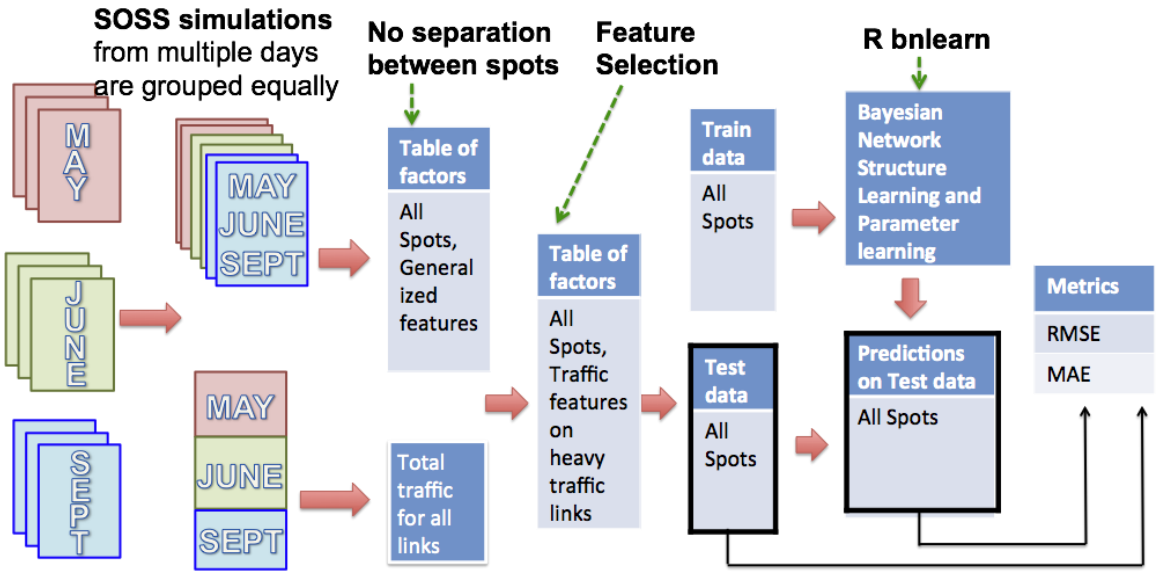


Figure 36. Processing Steps in the Enhanced Evaluation Framework to Model Aircraft Taxi-times for an Arbitrary Airport.

To identify the *high traffic links* for a given airport, we aggregate the simulation data, 1 file from each traffic day (see Figure 36; bottom of second column). This aggregated data contain all flights for the

given airport. We compute the total traffic in all links at the airport from the trajectories of all the flights in this aggregated data. We pick 50 of the links with the highest total traffic count as the *high traffic links*.

To enable training one BN model for one airport instead of one BN model for each spot (as in Phase I), we added an option to create table of factors with flights associated with all spots. This feature is also useful for making predictions for flights from different airports at the same time.

5.2.5 Learning Bayesian Network Model

Since we are using a generalized set of features to learn a BN model, the edge relations between variables are unknown. It is difficult to manually define edges between all BN variables as was done in Phase I because we identified 112 features from the Phase II training data. Hence we need to learn the structure and the parameters of the BN from the training data.

Several algorithms exist in the literature to learn BN structure from data [RN03]. Many of the algorithms rely on learning conditional independence relations between variables in the data. A commonly used approach is the Max-Min Hill-Climbing (MMHC) algorithm which is a hybrid of constraint-based and score-based structure learning techniques [TI06]. Although some improvements and extensions of this algorithm have been developed, reliable software implementing such methods is not available. Hence, we use the MMHC algorithm in our experiments.

The three transit-time variables (spot, merge and runway) have a temporal order. We provide this knowledge to the structure learning algorithm by using a set of blacklisted edges. The complete set can be summarized as follows:

- Runway must be a leaf node
- Merge can be parent of only runway (and not spot)

We use the *bnlearn* implementation of MMHC algorithm in our experiments. After the structure is learnt from data, we use the *bn.fit* method to learn the parameters of the conditional probability tables of all the variables. All the variables are modelled as the sum of Gaussian distributions.

5.2.6 Providing Mean and Standard Deviation for Predictions (bnlearn)

In order to generate multiple futures, we need to extract the probability distribution of the runway transit time variable. The variables are modeled as Gaussians, so we need the means and standard deviations of the posterior distributions. The *bnlearn* package provides the mean as the predicted value. However, to *predict* the runway arrival time of a departure, the method requires all parents of the runway variable to be present in the evidence set. At the time of prediction, if the aircraft has not yet arrived at the merge node, the values of spot and merge arrival time may not be known. We used a value of 0 for these cases. This does not affect the prediction, which is an inner product between the parents' values and coefficients for a linear Gaussian model.

During revisions of this method, we found a flaw in it. Setting absent parents to 0 does not yield the correct result. We need to fill in the missing values by applying stage-wise predictions. The times predicted for a departure to the spot and merge points in its 3D path should be used as evidences to predict the time of the departure to its runway. This approach, in effect, marginalizes the missing parents of runway arrival time and therefore provides an accurate prediction.

On the contrary, the standard deviations are not readily available from *bnlearn* package. We attempt to estimate them in two different approaches, CPDIST and an analytical method, described below.

5.2.6.1 CPDIST

Given a set of evidences, the *cpdist* method generates samples of an inquired variable using its conditional probabilities. From the samples we estimate the standard deviation of its posterior distribution.

However, this method requires significant computational time. As we need to make multiple predictions during scheduling, calling this method multiple times causes delay in the processing steps. Moreover, with evidences for many variables, this method often returns no sample. Hence we need to reduce the evidence set to obtain samples of an inquired variable.

Due to the above mentioned problems, we attempt to find an analytical form of the resulting conditional probability distribution of the runway variable.

5.2.6.2 Analytical Method

The variance of a sum of independent Gaussian variables is equal to the sum of the individual variances [KD09]. For flights in the scheduling process, the unobserved variables (4D trajectory times to be predicted) can be those at the spot, the merge point, or none (i.e., the departure has passed the spot and merge point, so the times are known). Given the evidences, the variance of the runway node will be the sum of its own variance and the variances of the other unobserved nodes in the BN. Using this fact we estimate the standard deviations of gate to runway arrival times.

This approach is very fast as it requires only one dot product computation (between an indicator vector of unobserved variables and vector of variances). Moreover, unlike CPDIST, we always can use the full set of evidences.

5.2.7 Experiments and Results

This section presents the experiments and results with the generalized taxi time modelling approach. In all experiments, we use the Phase II data which includes the SOSS simulation data from three traffic days. We start with comparing the Phase I and Phase II models. Then we investigate the effect of increasing training samples on the Phase II model performance. Near the end of this section, we present the prediction performance on all airports.

5.2.7.1 Comparison to Phase I Prediction Accuracy

Here we compare the Phase I and Phase II methods using the Phase II data from JFK airport. Since creating a TOF from all the SOSS simulation data requires much more computational time, we used data from 100 simulations (100 files) for each traffic day, resulting in a dataset capturing data from 300 different simulations (100 simulations for each airport). The TOF entries created from all these simulations are split into training and test sets using a 60:40 ratio.

Figure 37 shows the learned structure of MMHC algorithm with a few important nodes and edges enlarged. The network has 112 nodes and 1693 edges.

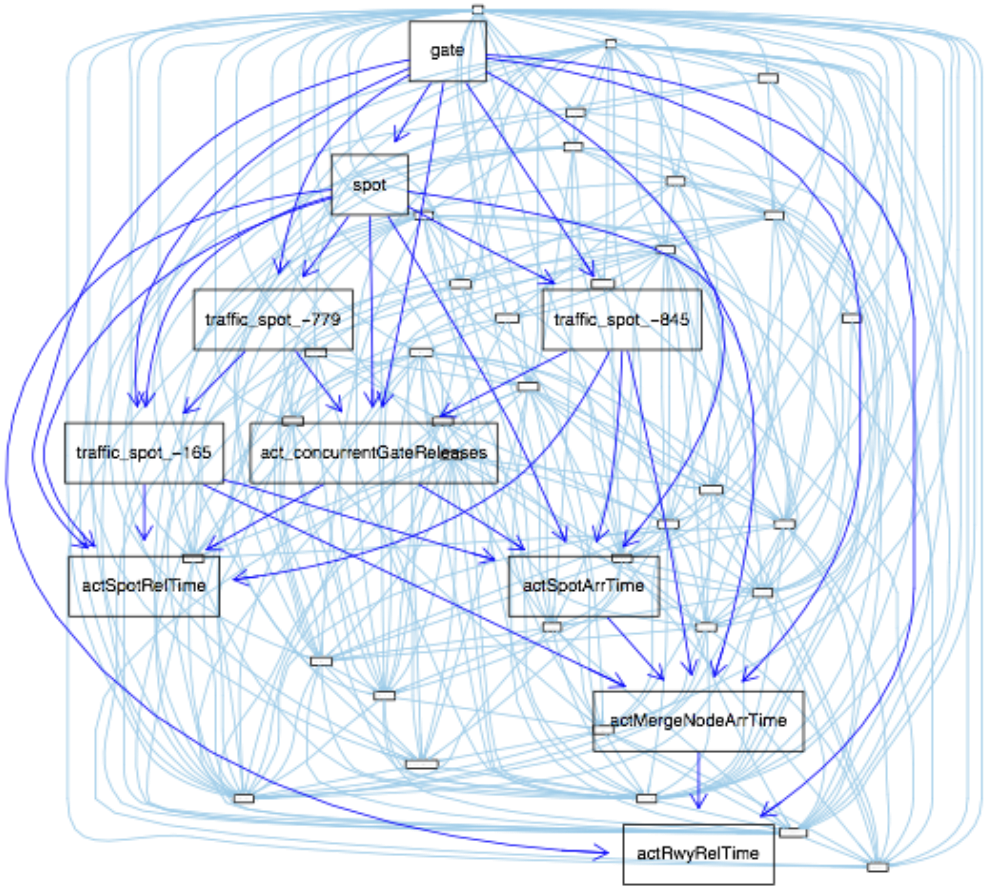
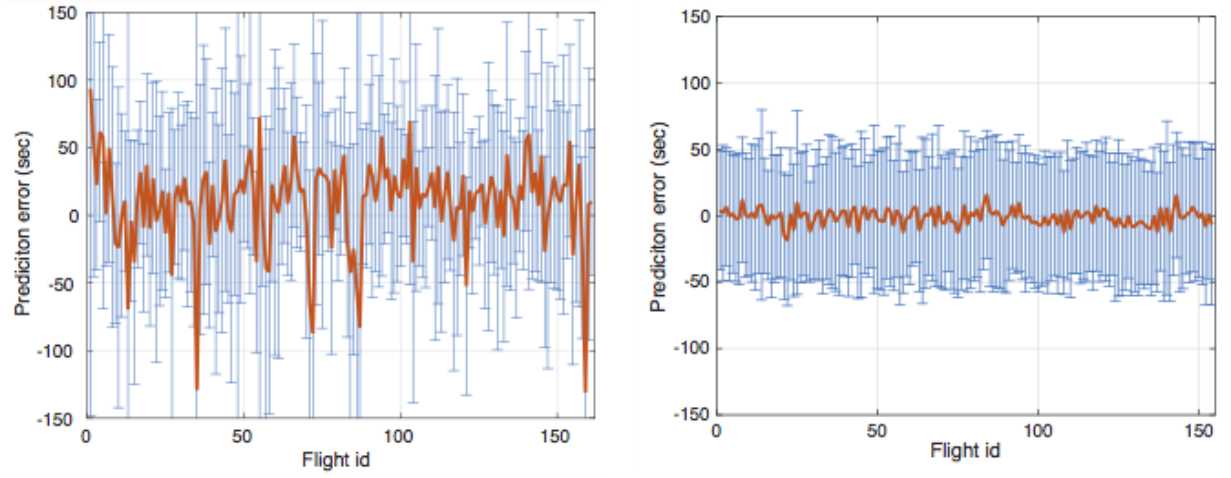


Figure 37. BN Structure For High Traffic Nodes Learned from JFK Simulation Data Using the MMHC Algorithm

Figure 38 presents a comparative view of the prediction errors for gate to runway transit times of two BN models created using the Phase I and Phase II approaches.



a) Phase I model **b) Phase II model**

Figure 38. Comparison of the Phase I (Left) and Phase II (Right) Mean and Standard Deviations of Prediction Errors for JFK Gate to Runway Transit Times

The MAE and RMSE of the transit time prediction errors of the Phase I and Phase II models are:

Phase I model: MAE = 61.5 seconds, RMSE = 98.61 seconds

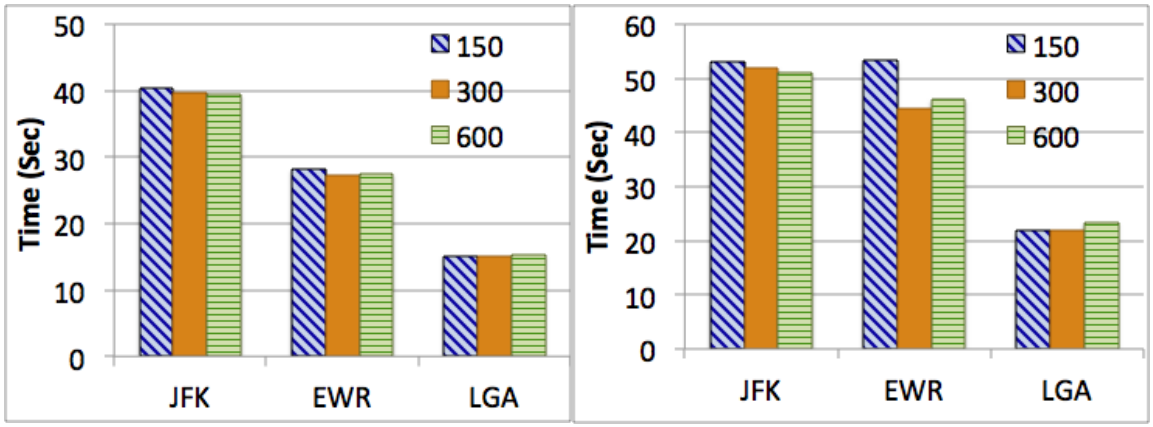
Phase II model: MAE = 39.4 seconds, RMSE = 51.1 seconds

The results indicate the prediction performance is improved when using the Phase II model.

5.2.7.2 Sensitivity of Prediction Accuracy to Quantity of Training Data

To train models in a reasonable amount of processing time, we used only a subset of the simulations. However, one may ask, will increasing training data improve performance? We investigate this question by increasing the quantity of data used for the training data set from 60 percent of records in 150 files to from 60 percent of records in 600 files and comparing the prediction errors of the trained models. In this experiment, we select gate-runway transit time as the variable of interest and flights through one spot.

Figure 39 shows the comparison of MAE and RMSE for all three airports with increasing number of training samples. The values are tabulated in Table 4.



a) Mean absolute error **b) Root mean squared error**

Figure 39. Prediction Errors of Phase II Approach with Increasing Training Samples from Flight Trajectories Through one Spot

Table 4. Prediction Errors of Phase II Approach with Increasing Training Samples from Flight Trajectories Through one or ALL Spots

		Mean Absolute Error (MAE), Seconds			Root Mean Square Error (RMSE), Seconds		
# of spots	Size	JFK	EWR	LGA	JFK	EWR	LGA
1	150	40.41	28.10	15.13	52.94	53.29	21.80
1	300	39.70	27.25	15.12	51.85	44.44	22.00
1	600	39.38	27.43	15.22	51.07	46.01	23.29
ALL	150	64.99	53.59	21.93	85.73	76.00	50.00

In almost all cases, the change in prediction error is minor. And the performance for 300 and 600 is very similar. Hence, we conclude that we need not use large number of simulation files to obtain superior models. Finally, we present the prediction results from a model trained and tested with flights through all spots. The last row of Table 4 shows the errors for all three airports. We notice that the maximum error is about 1.5 minutes.

5.2.8 Simultaneous Application to Three Airports

We established a method to predict runway arrival times for a departing flight based on its transit time probability distribution. The transit time prediction framework is integrated with the PROCAST scheduler to schedule flights for multiple airports simultaneously. This requires predicting taxi times for flights originating from different airports.

We start with training three different BN models for three airports. The training sets are created using flights through all spots. We save the model parameters and the feature information as static data. During scheduling cycles, feature values are computed on the fly and transformed into appropriate data structures for BN predictions. Each flight, depending on its airport-id, is predicted using the appropriate BN model. The predicted mean and standard deviations are passed to PROCAST.

6 Enhancements to SOSS

In Phase II of the project we enhanced the NASA SOSS fast-time simulation with models of multiple airports and the terminal airspace of the New York metroplex, the ability to simultaneously simulate multiple airports, and the ability to schedule arrival and departure traffic of multiple airports. We describe the numerous components to the New York metroplex model, including airport runway configurations, airport surface and terminal airspace modeling, separation rules, traffic scenarios and validation of the metroplex model (Section 6.1). We describe enhancements to the SOSS simulation software enabling

simulation of New York metroplex traffic and scheduling of multi-airport arrival-departure traffic simulated by SOSS.

6.1 New York Metroplex Model

In Phase II we developed link-node models of the surface and (limited) terminal area for the other primary airports in the New York metroplex, EWR and LGA, and implemented them in SOSS. The airport model generation process involves the following steps: 1) selecting the airport runway configuration, 2) generating the required airport adaptation data, and 3) generating traffic scenarios. For the sake of completeness we include descriptions of the airport adaptation data such as the link-node models of the airport surface and terminal routes that were previously developed for JFK as well. Additional details regarding the development of the SOSS model of JFK can be found in prior reports [SS12] [SS15].

6.1.1 Airport Runway Configurations

In the New York metroplex, the runway configurations used at the three major airports are coordinated by air traffic managers. We use runway configurations for EWR and LGA based on the JFK runway configuration used in Phase I: arrivals using both runways 31L and 31R, and departures using runway 31L. We analyzed FAA Aviation System Performance Metrics (ASPM) data for the time period of January 2015 to November 2015 to determine the most commonly used runway configurations for the other New York area airports when JFK is running 31L, 31R|31L. Figure 40 shows the results of this analysis. At LGA the configuration with arrivals using runway 31 and departures using runway 04 (31|04) was used 25% of the time when JFK was using 31L, 31R|31L. The EWR configuration most commonly used during this period, at 42% of the time, was arrivals on runway 04R and departures on runway 04L (04R|04L).

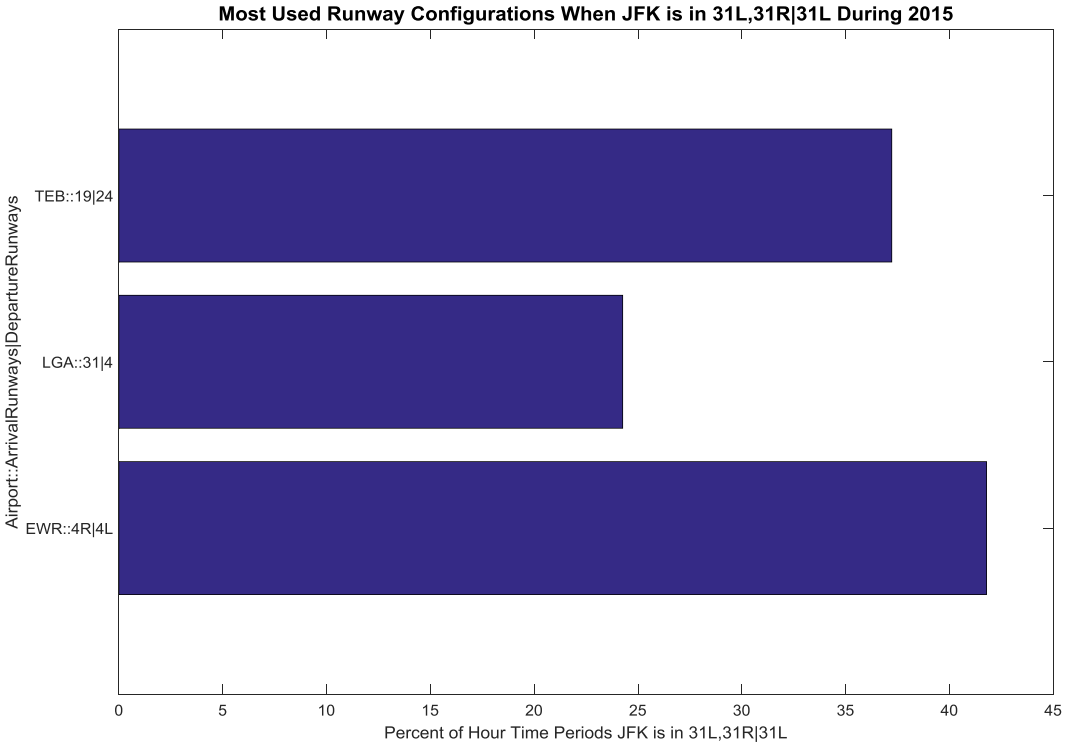


Figure 40. Analysis of ASPM Data to Determine Runway Configuration Models for EWR and LGA

The FAA has a useful website[2] that allows visualization of interacting arrival flows, airspace, and fixes for various runway configurations for the New York area airports. The joint runway configurations and route structure that we model is based on a configuration shown in Figure 41, which is from the FAA TFM Learning Website.

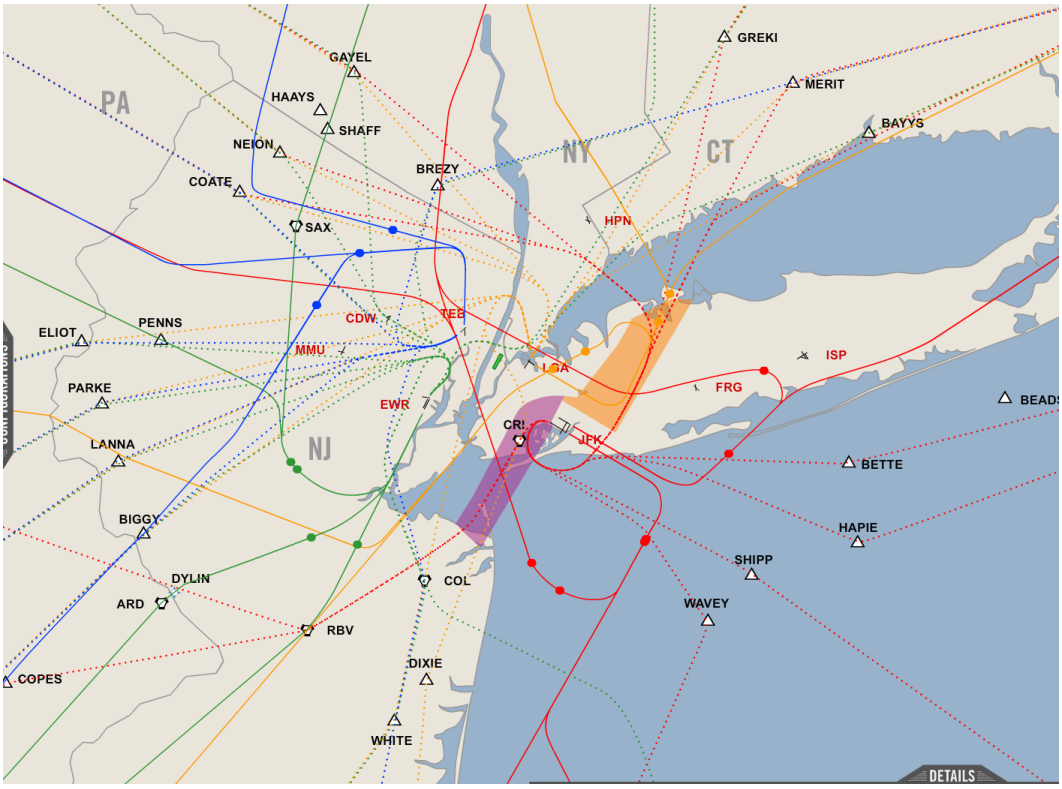


Figure 41. Joint Runway Configurations and Route Structure for JFK, EWR, and LGA

6.1.2 Surface Link-Node Models

The methods used to generate the JFK link-node model form the basis for our development of the models for EWR and LGA under this research effort. The JFK SOSS surface link-node model used in Phase I was originally developed in 2012 using the methods documented in [SS12]. For EWR and LGA the team started with surface link-node models that were originally generated for the NASA VAMS SLDAST project [AS09] [VC09]. We first convert the link-node model to SOSS format resulting in a two-dimensional representation of the airport surface in SOSS. The link-node model is generated by adding and defining the node types in SOSS including spot nodes, departure nodes, arrival nodes, queue nodes, runway crossing nodes, and any additional intersection nodes. In addition we define the runway geometry by capturing the *x* and *y* coordinates of the vertices and by defining the length of the runway in meters. The runway heading is defined based on the FAA airport diagram and accounts for magnetic variation. The following figures show the surface link-node models along with a tabulation of the surface link and node types for EWR, LGA, and JFK.

[2] http://tfmlearning.faa.gov/NY_Airspace/NY_Airspace_Pkg/NY_Airspace.swf

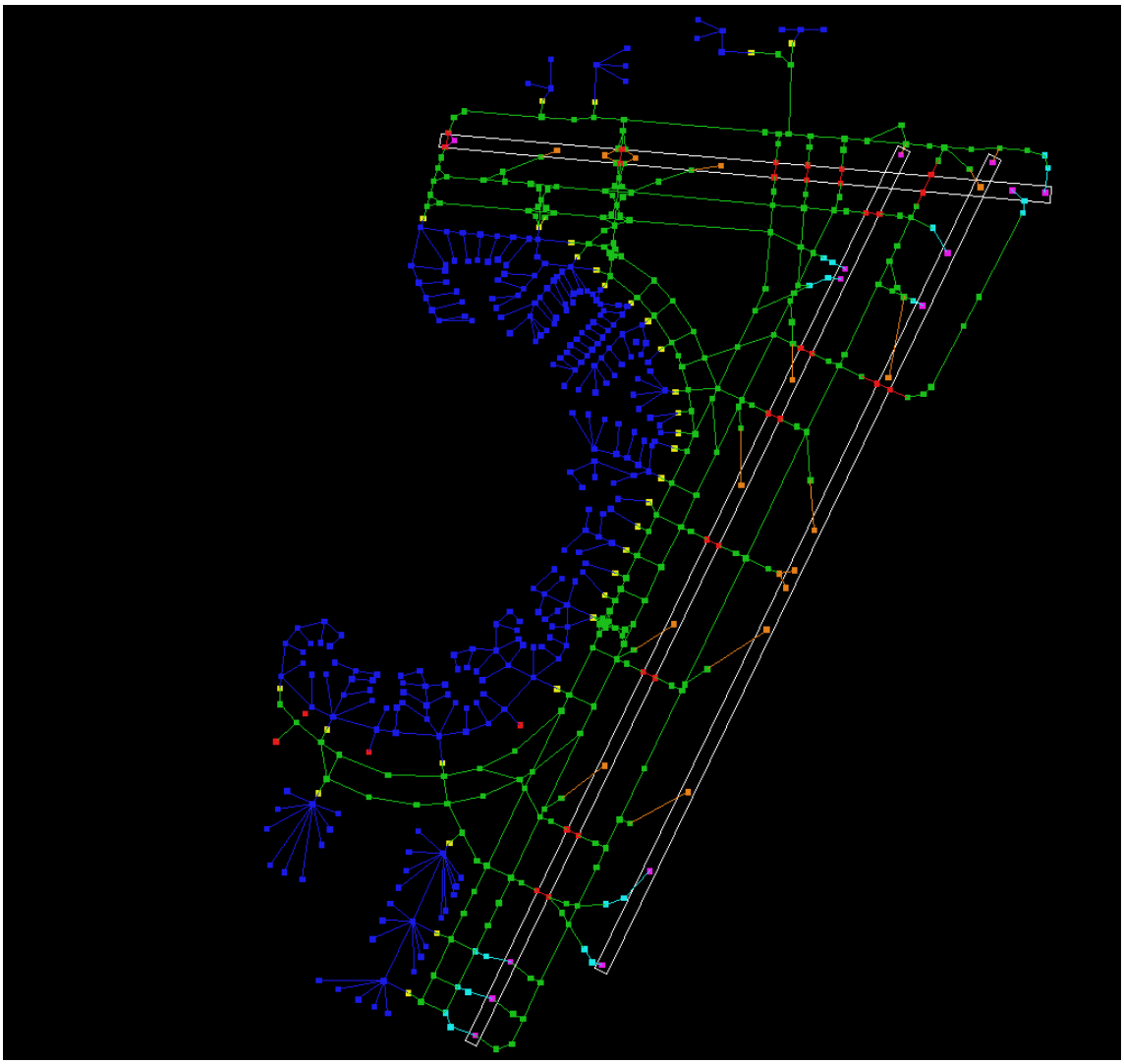


Figure 42. EWR Link-Node Model as Implemented in SOSS

Table 5. EWR Link Node Model Characteristics

Node Type	Quantity	Link Type	Quantity	Color Identifier
Arrival	20	Arrival	19	Orange
Departure	14	Departure	2	Magenta
Queue	20	Queue	157	Cyan
Runway Crossing	28	Runway Crossing	18	Red
Taxiway	225	Taxiway	356	Green
Spot	32	Spot	1	Yellow
Ramp	99	Ramp	356	Blue
Gate	158	Gate	18	Blue

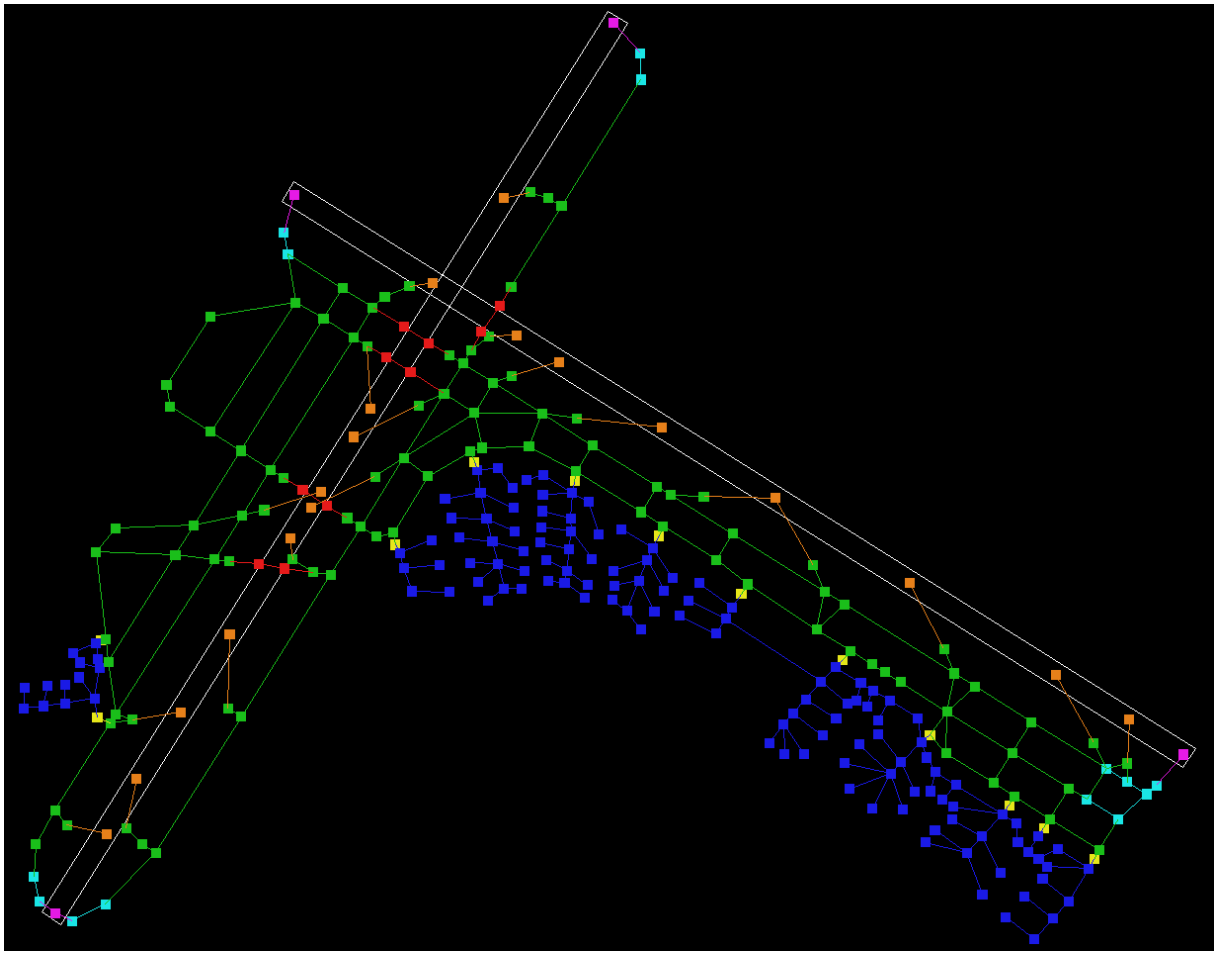


Figure 43. LGA Link-Node Model as Implemented in SOSS

Table 6. LGA Link Node Model Characteristics

Node Type	Quantity	Link Type	Quantity	Color Identifier
Arrival	18	Arrival	19	Orange
Departure	4	Departure	5	Magenta
Queue	14	Queue	9	Cyan
Runway Crossing	10	Runway Crossing	15	Red
Taxiway	96	Taxiway	141	Green
Spot	12	Spot	1	Yellow
Ramp	57	Ramp	61	Blue
Gate	74	Gate	74	Blue

Figure 44. JFK Link-Node Model as Implemented in SOSS

Table 7. JFK Link-Node Model Characteristics

Node Type	Quantity	Link Type	Quantity	Color Identifier
Arrival	20	Arrival	23	Orange
Departure	15	Departure	17	Magenta
Queue	72	Queue	60	Cyan
Runway Crossing	36	Runway Crossing	13	Red
Taxiway	481	Taxiway	618	Green
Spot	26	Spot	37	Yellow
Ramp	198	Ramp	236	Blue
Gate	121	Gate	121	Blue

6.1.3 Terminal Procedures

In Phase I of this research effort we made enhancements to the SOSS platform to include modeling of arrival and departure routes for the runway configuration modeled at JFK [SS14] [SS15]. Terminal

procedures for JFK were derived from recorded surveillance data. Arrival routes from each arrival fix to each runway, and departure routes from each runway to each departure fix were determined from examination of surveillance track data. The surveillance track data was recorded by the Saab Sensis Aerobahn system operating at JFK. This system records aircraft position data on the airport surface and in the terminal airspace. Data used for this effort were recorded during the Fall of 2013. For each fix-runway or runway-fix pair, route clusters within the surveillance data were identified and less common flight paths were filtered out. The clusters were compared to arrival and departure procedures published for JFK, and the procedure that most closely matched the route cluster was used as the basis for the modeled Terminal Procedure route. The modeled Terminal Procedure routes included fixes identified in the published procedure as well as intermediate fixes added to define to the route where needed. This process produced a modeled route as a sequence of latitude-longitude waypoints extending from arrival fix to runway or from runway to departure fix [SS15]. We follow a similar process to define the arrival and departure routes for the EWR and LGA in Phase II. In addition to using ASDE-X data, we use CTAS *cm_sim* data [ME14] provided by NASA along with published charts and Architecture Technology's AvScenario [SD04] scenario generation tool to identify commonly used arrival and departure routes. As in Phase I, we develop a set of routes for the given runway configuration that is modeled as a sequence of waypoints from runway to fix, and fix to runway.

6.1.4 Arrival and Departure Fixes

The team identified the arrival and departure fixes for the EWR, LGA and JFK airport model using multiple online resources such as AirNav.com and FAA OIS. Table 8 lists the modeled arrival and departure fixes for each of the airports.

Table 8. Modeled Arrival and Departure Fixes for EWR, LGA, and JFK Airports

Airport	Arrival Fixes	Departure Fixes
EWR	SAX, PENNS, DYLIN, RBV	BAYYS, BIGGY, BREZY, COATE, COL, ELIOT, GAYEL, GREKI, LANNA, NEION, PARKE, WHITE
LGA	EMPYR, HAARP	BAYYS, BIGGY, COATE, COL, DIXIE, ELIOT, GAYEL, GREKI, LANNA, MERIT, NEION, PARKE, WHITE
JFK	CAMRN, LENDY, ROBER	BDR, BETTE, COATE, DIXIE, GAYEL, GREKI, HAPIE, MERIT, NEION, RBV, SHIPP, WAVEY

6.1.5 Surface Routes

The team generated a list of surface arrival and departure routes, whereby each route is defined by specifying a sequence of nodes between gates and runways. A surface arrival route starts with an arrival node at the runway and ends with a gate node. A surface departure route starts with a gate node and ends with a departure node at the runway. For EWR and LGA we specify gate-departure node pairs, arrival node-gate pairs, and intermediate nodes for each pair. We then use the "shortest path" feature in SOSS to define taxi-in and taxi-out routes for all possible node pairs. This process results in 833 arrival and departure routes for EWR, and 444 arrival and departure routes for LGA. During our validation of the

surface models, it was noted that the LGA shortest path taxi routes resulted in deadlock situations in SOSS. This meant that SOSS was in some cases unable to resolve potential conflicts between aircraft, and would not be able to complete the simulation run. Because of this we modified the taxi-in and taxi-out routes to reflect a more realistic use of the taxiways that also avoided the deadlock situation in SOSS. The JFK predefined arrival and departure taxi routes were generated based on analysis of surface surveillance data [SS12]. In the JFK model, there were 2 arrival runways (e.g., 31L and 31R), 1 departure runway (e.g., 31L), and 183 gates. There were 6 runway exits (3 each for 31L and 31R) among the arrival runways, and 3 runway entry points for the departure runway. The various gate-runway entry/exit combinations yielded a total of 2,460 routes for JFK for just a single runway configuration. This included 1,230 arrival routes and 1,230 departure routes.

6.1.6 Runway Separation Requirements

In SOSS, separation matrices are used to specify the minimum required temporal separation between two successive operations on the same runway or a pair of interacting runways in the simulation. Matrices define different separations for different leader-follower aircraft weight classes. The separation matrices cover four aircraft weight classes—Heavy (H), Large (L), Boeing 757 (B757), and Small (S) as per the standard FAA definitions [SS12]. For each modeled airport runway configuration, we define the possible interactions between aircraft arrivals and departures. Runway interactions are based on factors such as runway geometry (e.g. same runway or parallel runways) and/or pairs of aircraft operations (e.g. arrival 1 causes wake turbulence for departure 2). As a result, a single airport and operating condition may require multiple separation tables to describe the minimum separation requirements for the specified runway configuration [ACES05]. Note that because SOSS does not delay arrivals, no separation matrix is specified for arrival-arrival pairs. For JFK, we use the runway interactions and runway separation matrices defined in [SS12]. For EWR and LGA we use runway separation matrices that correspond to the runway interactions from the individual runway modeling data developed during the NASA ACES B3 development effort [ACES05]. The runway interactions and actual values for the runway separation matrices are contained in Appendix A of this document.

6.1.7 Traffic Scenarios

Traffic scenarios are generated based on historical NAS-wide demand data from 2011-2012 provided to NASA by the FAA Air Traffic Organization (ATO). These same sample days were previously used for a NASA project [LMI15] focused on analyzing and identifying choke points in the NAS. The FAA generates the schedule for a set of 16 sample days for each calendar year. Four sample days are selected in each quarter to stratify representative different demand levels, seasonal patterns, weather patterns, and delay outcomes across the NAS. The days are selected so that their averages match to the annual averages for the entire NAS and for the Air Traffic Control Centers in major statistics including demand level, delay for all flights and for IFR flights [LMI14]. The 16 samples days provided by the FAA are: 10/1/2011, 10/30/2011, 11/17,2011, 12/19/2011, 2/04/2012, 2/28/2012, 3/16/2012, 3/20/2012, 4/30/2012, 5/04/2012, 5/13/2012, 6/11/2012, 7/14/2012, 7/25/2012, 7/30/2012, 9/05/2012. The availability of 16 sample days can be used to support annualization of results by capturing a variety of influences in the sample. In addition, the use of sample days of different demand levels, seasons, and weather can be used to support sensitivity analysis. Figure 45 shows the number of ASPM operations for each sample day. This gives an indication of traffic levels in each of the sample days [LMI15].

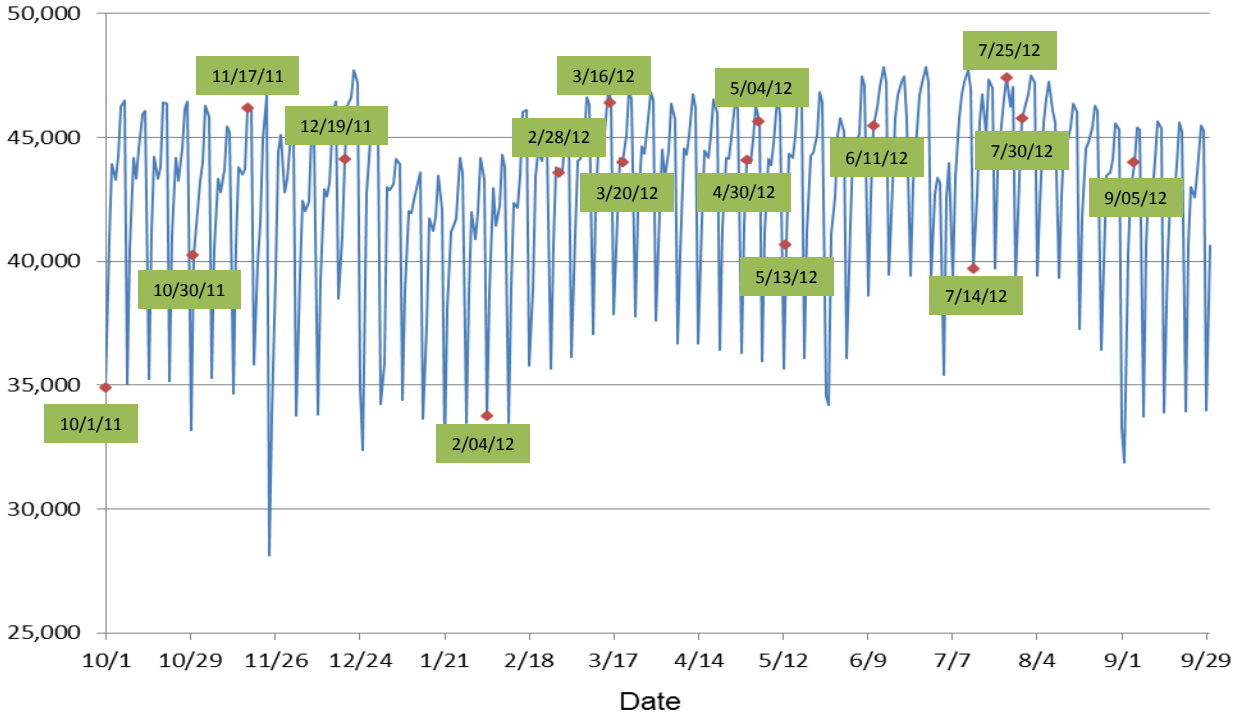


Figure 45. FAA ATO Sample Days, Labeled in Green, Provided to NASA for this Research (Courtesy of LMI)

The FAA demand sets contain ETMS-based information on flights operating in the NAS such as flight plan information as well as planned/reported Gate Out, Wheels Off, Wheels On, and Gate In times [FAADD]. However to make traffic scenarios for use in SOSS we need to supplement the FAA demand data with all of the per flight information required for a traffic scenario in SOSS. We first identify the required data fields for SOSS traffic scenarios, and then determine the data sources for those fields. The required data includes gate assignments, spots, runways, departure and arrival fixes, departure and arrival routes, and departure and arrival times. For gate assignments we analyze FlightStats data to characterize distribution of airlines to gates, and then develop an algorithm to assign flights to terminals. We leverage gates defined in SOSS adaptation data for each airport; and assign individual gates to flights in each terminal using a round-robin distribution method. We assign spots to flights based on spot-terminal mapping inherent in SOSS adaptation data for airport. Runway assignment for departures is straightforward since we have only one departure runway for each airport. For multiple arrival runways (JFK), we use a method that we developed for the Choke Point project [LMI15] to assign runway based on origin airport. Similarly, we use methods to assign departure or arrival fix based on origin (arrivals) or destination (departures) airport. The departure route is unique to the runway-fix combination. We determine the simulation start time for departures based on the scheduled gate departure time, and the start time for arrivals as the actual or scheduled "On" time minus the transit time. We obtain this information from the SOSS adaptation data, the FAA traffic files, and other external sources. We then determine the methods for assigning data to each field for each flight and develop scripts that:

- Read in FAA flight traffic demand set, extract flights of interest, put each flight into SOSS-compatible data structure, and populate each flight's data structure fields with data available.
- Read SOSS adaptation data and based on developed rules, assign each flight's data structure fields with data available.

- Based on external data sources and developed rules, assign each flight's data structure fields with data available.
- Write output file as per SOSS format requirements.
- Conduct SOSS simulation to verify compatibility.
- Analyze SOSS output data to verify flights are being simulated as per input file specifications/data.

In this manner we develop SOSS traffic scenario files that will provide simulated traffic for the three airport surface and terminal models implemented for EWR, JFK, and LGA. These methods could potentially be applied in future research to develop SOSS traffic scenarios for any modeled SOSS airport surface and terminal airspace model.

We selected a subset of the 16 sample days to generate traffic scenarios for validation of the SOSS models, training of the BNs, and evaluation of our PROCAST solution architecture.

Table 9. Traffic Scenarios for Phase II Training, Validation and Evaluation

	EWR	**JFK**	**LGA**
Training	9/5/12 18-23 Local 7/25/12 6-11 Local	5/13/12 2-7 Local 3/16/12 13-16 Local	9/5/12 18-23 Local 7/25/12 6-11 Local
Validation	5/13/12 0-23 Local 6/11/12 0-23 Local 9/5/12 0-23 Local	5/13/12 0-23 Local 6/11/12 0-23 Local 9/5/12 0-23 Local	5/13/12 0-23 Local 6/11/12 0-23 Local 9/5/12 0-23 Local
Evaluation	7/25/12 7-9 Local 3/16/12 8-10 Local	7/25/12 7-9 Local 3/16/12 8-10 Local	7/25/12 7-9 Local 3/16/12 8-10 Local

6.1.8 Model Validation

Airport surface models are validated by comparing simulated operational metrics such as taxi-in and taxi-out times with historical operational metrics from the FAA's ASPM performance database. For SOSS validation, we developed traffic scenarios for time periods during which the airports were actually operating with the runway configurations used in our SOSS adaptations. We then compare SOSS traffic counts and taxi-in/taxi-out times with FAA ASPM data for those time periods of interest.

Table 10 shows a comparison of the arrival and departure counts simulated in SOSS for our validation traffic scenarios with the historical traffic counts contained in the FAA ASPM database. The SOSS input file creation software captures arrivals and departures by their scheduled arrival and departure times in the FAA data. We verified that number of flights reported in the SOSS simulation approximate the number of flights reported in the SOSS simulation analysis results. However, there may be differences in the number of SOSS simulated flights and the ASPM scheduled flights. This FAA schedule data may be different from the schedule data used for the ASPM counts due to changes in the scheduled flights throughout the day including the actual number of flights in the schedule and when scheduled arrival and departure times are binned.

Table 10. Comparison of SOSS and ASPM Arrival and Departure Counts

			Total Departure Count		Total Arrival Count	
Airport	**Date**	**Time**	**ASPM**	**SOSS**	**ASPM**	**SOSS**
EWR	9/5/2012	18-23 Local	151	161	177	151
EWR	7/25/2012	6-11 Local	213	231	149	170
JFK	5/13/2012	2-7 Local	47	58	62	57
JFK	3/16/2012	13-16 Local	111	122	166	168
LGA	9/5/2012	18-23 Local	127	120	184	142
LGA	7/25/2012	6-11 Local	234	241	169	180

We originally used the shortest-path taxi routes that are automatically generated by SOSS based on the airport link/node model. However, during our validation activities we observed that the traffic scenarios and the routes used led to cases of "gridlock" in SOSS where SOSS would not be able to resolve conflicts between taxiing aircraft. For this reason we modified the taxi routes modeled at LGA to try to avoid the gridlock situation – particularly between arriving and departing traffic. After making this modification we reran our SOSS validation traffic scenarios in order to compare taxi times in SOSS with historical taxi-time data from ASPM. Table 11 shows that SOSS is fairly consistent in the times it simulates, but both the taxi-out and taxi-in times are low compared to ASPM. Although taxi-times do not match historical data, they are consistent and therefore provide a reasonable basis for evaluating different models of uncertainty and prediction.

Table 11. Comparison of SOSS and ASPM Taxi-Out and Taxi-In Times

			Average Taxi-Out Time (minutes)		Average Taxi-In Time (minutes)	
Airport	**Date**	**Time**	**ASPM**	**SOSS**	**ASPM**	**SOSS**
EWR	9/5/2012	18-23 Local	25.2	6.2	9.7	6.1
EWR	7/25/2012	6-11 Local	27.6	6.2	8.7	5.9
JFK	5/13/2012	2-7 Local	17.6	10.1	7.1	5.8
JFK	3/16/2012	13-16 Local	33.2	10.5	8.1	6.5
LGA	9/5/2012	18-23 Local	27.9	5.1	9.5	4.1
LGA	7/25/2012	6-11 Local	30.7	5.1	8.1	4.3

We reviewed and discussed our modeled taxi routes and our traffic counts and taxi-time comparisons with Ralph Tamburro of the PANYNJ. Mr. Tamburro offered the following comments regarding our taxi route models and our validation results.

- The taxi routes for EWR and LGA looked reasonable.

- The taxi-out differences are most likely due to flow restrictions impacting departures that SOSS is not modeling.
- JFK departure metering was in operation during 2012. Metering accounts for the runway departure rate but not the departure restrictions. Metering attempts to maintain a target queue length of 12-15 departures. JFK departure metering specifies movement area entry time. Aircraft OUT times to meet movement area entry times can vary: departures can push back early to absorb delay in the ramp, or push back later to absorb delay at the gate.
- Also, at LGA, uncertainty in arrival landing times impacts departure throughput due to the crossing runways. This contributes to taxi-out delay. The arrivals to runway 31, departures from runway 4 configuration is particularly sensitive to this because of the circling approach to runway 31, and the intersection of the runways is at the far (long) end of the runways.
- Regarding the number of flights:
    - EWR: 213/231 departures for 7/25/2012 is normal; 151/161 departures for 9/5/2012 is low.
    - LGA: 127/120 departures for 9/5/2012 is light; 234/241 departures for 7/25/2012 is low.
    - JFK: 111/122 departures for 3/16/2012 3/16/2012 seems normal; 47/58 departures for 5/13/2012 is low. JFK weather on 3/16/2012 looked okay. IFR day, but winds mild, visibility cleared in late afternoon.

6.2 Multi-Airport Traffic Scheduling

The operational environment for the multi-airport simulation involves multiple SOSS processes—one simulating each metroplex airport—and the multi-airport scheduler (see Figure 46). This is enabled by a socket interface in the scheduler that creates an individual socket for each SOSS process. The scheduler acts as a server and waits for the SOSS processes. The SOSS processes call the scheduler periodically based on a set call interval (e.g. every 300 seconds of simulation time) and send the scheduler a data package of flights to be processed within the particular schedule cycle.

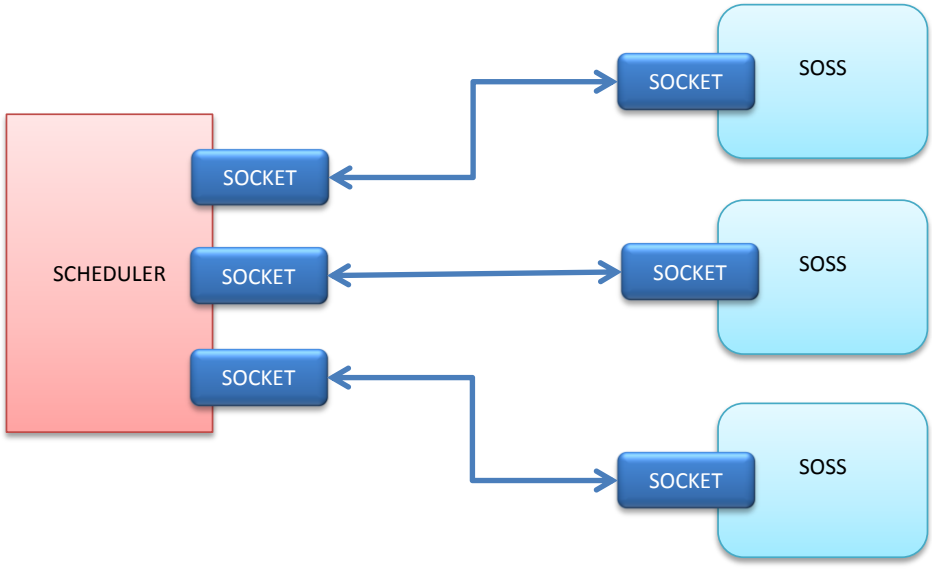


Figure 46. Scheduler-SOSS Architecture

6.2.1 Synchronization

The individual SOSS applications have no capacity to stay synchronized with each other, so the scheduler must synchronize the simulations. To do this, the scheduler is designed to wait for each SOSS process to complete its cycle prior to actually processing the data. In-between schedule cycles, each SOSS process runs independently and at its own pace. The SOSS applications run with the same call interval and to "Block" the scheduler calls so that the SOSS processes don't start running again until a scheduler message has been received. By blocking on scheduler calls, maintaining the same call interval, and by the scheduler waiting for each SOSS to return before scheduling, the simulation stays synchronized.

There are two additional modifications to SOSS to make the metroplex simulation function properly. First, SOSS needed to always send a message and block on the call interval regardless of whether there were any flights in the forecast window. The nominal version of SOSS only sent a message if there were flights to schedule so that if there was a gap in the demand set, one SOSS could perpetually suspend the entire simulation. To overcome this, a special empty message protocol was set up between the scheduler and SOSS so that SOSS would always send message and block regardless of message content. The other problem occurred when a particular SOSS process finished its run prior to the other SOSS processes and would stop sending messages. This would cause the simulation to perpetually suspend as well. The SOSS code was modified to continue to run after completing the simulation of its own traffic demand set, and to send empty messages until a prescribed time was reached.

6.2.2 Pushback Times

Each SOSS process sends a data package to the scheduler that includes information about all the flights within the forecast window. This information includes the routing of the aircraft and the originally scheduled pushback time (departure) or its arrival fix crossing time (arrival). When the scheduler sends back a de-conflicted schedule it sends back updated pushback times. However, on successive cycles, SOSS doesn't send the updated pushback time to the scheduler, but rather the original pushback time, even though SOSS will apply the updated pushback time when releasing the aircraft. Since the scheduler does not maintain state between cycles, the actual pushback time is unknown to the scheduler after the first scheduling of a particular flight. Furthermore, SOSS seems to always use the latest push time (the one most in the future) rather than the push time from most recent message received. Therefore, the current schedule cycle could not override prior push times unless it was adding additional delay. To correct this, the SOSS message was modified to send back the delayed pushback time in addition to the original pushback time. This way, the scheduler knows the actual push back time for a particular flight. This additional message element is illustrated in Figure 47.

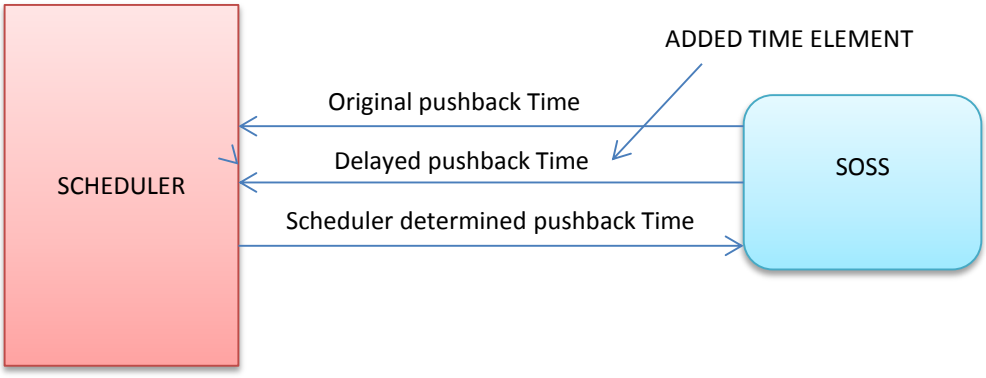


Figure 47. Scheduler-SOSS Message Time Fields

The fast-time simulation experiments with SOSS evaluated a baseline scheduler and the PROCAST probabilistic scheduler. The baseline scheduler was modified to use the delayed pushback time, anytime it was later (in the future) than the original time. When running the baseline scheduler alone, this improved

the scheduler performance from a conflict reduction perspective. Virtually no spacing conflicts were observed in this configuration. Unfortunately, when the baseline scheduler was run in this configuration in the multi-future context of the PROCAST probabilistic scheduler, the use of the delayed pushback time led to cascading delay instability. That is, delay would be continually added and some flights would never finish. This phenomenon is still not completely understood. To correct this problem, the use of the delayed pushback time had to be suspended. Instead, the scheduler would never use anything other than the original pushback time. This fix results in a less precise scheduler, where not all aircraft are scheduled to be in a conflict-free state, however it lacks any cascading delay instability.

7 Evaluation of PROCAST Using Fast-Time Simulation

This section presents the design of experiments for and results and observations from simulation-based evaluation of the PROCAST multi-airport traffic scheduling with deterministic and alternative probabilistic departure taxi time models. In addition, alternative methods for specifying a traffic schedule from multiple scheduled futures are also explored. We present the experiment design, including the evaluation architecture, metrics, PROCAST control variables and traffic conditions (Section 7.1). We present the departure delay results obtained from the simulation evaluations of the PROCAST conditions for the New York metroplex (Section 7.2). We present the methods for and results of evaluating alternative methods for PROCAST to specify the "statistically best" traffic schedule (Section 7.3).

7.1 Experiment Design

In Phase II we focused on developing and investigating the use of BNs to probabilistically model and predict taxi-times on the airport surface within the PROCAST framework. After completing integration and testing of the major components of the PROCAST solution architecture including the SOSS airport surface/terminal models, the multi-airport scheduling algorithms, and the BN evaluation framework and engine, we ran a number of traffic scenarios. We compared the effectiveness of the BNs with a simple probability model (distribution). We also compared the effectiveness of the probability models based on the number of futures used in a given scheduling cycle. The hypothesis was that the use of probabilistic modeling for taxi-time prediction makes the integrated arrival and departure scheduling more robust to uncertainties and therefore more likely to be useful in future air traffic management decision support tools.

Table 13 shows our experiment matrix describing the PROCAST runs along with the various probability modeling configurations. We made runs of PROCAST using two-hour traffic scenarios derived from the FAA demand data that are dissimilar to the traffic scenarios used for training the BNs. Simulation with interacting traffic causes sensitivity to initial conditions (gate pushback times = pushback readiness times + scheduling delays), and therefore uncertainty in taxi times.

7.1.1 Test Architecture

Fast-time simulation-based assessments were conducted using three synchronized instances of the SOSS simulation software, each simulating a different airport in the New York metroplex, as described in Section 6.2. In this configuration, the CD&R feature of SOSS is turned on. Using this platform, we tested and compared the performance of metroplex airport traffic under the control of either a baseline scheduler or the PROCAST probabilistic scheduler. The baseline scheduler for the evaluations used the multi-airport scheduling algorithms of PROCAST described in Section 4.1, however only scheduled a single future predicted using purely deterministic transit time models. The deterministic transit time models correspond to unimpeded transit of an aircraft as modeled by SOSS. The PROCAST probabilistic scheduler for the evaluations used the multi-airport and probabilistic scheduling components described in Section 4. For the evaluations, the PROCAST scheduler used one of two probability models for gate-to-runway taxi time: A *simple* model or a BN model.

The *simple* probability model approximates the distribution of departure taxi times for each airport as Gaussian with mean and standard deviation that are fixed. The values used are given in Table 12.

Table 12. Parameters for the Simple Probability Model for each Airport

Airport	Mean (seconds)	Standard Deviation
John F. Kennedy (JFK)	670.5	266.4
Newark (EWR)	385.9	124.8
LaGuardia (LGA)	326.6	83.2

These parameters were extracted from the Phase II simulation data that we also used to train and test the Bayesian network models. This data set is described in Section 5. We used the whole data set, and straightforwardly computed these two statistics for all departures, per airport.

The BN models of departure taxi times for the airports were created as per the methodology described in Section 5. Each model treats the gate-to-runway taxi time as a random variable in a (large) joint distribution. The other variables in the distribution include features of the individual flight being predicted, as well as metrics of the traffic level at many significant points on the airport surface. The model uses Bayesian inference to compute a posterior distribution for taxi time, given the measured or predicted values of most of the other variables. We used machine learning to derive the joint distribution, in the form of a Bayesian network, from extensive set of SOSS simulation output data for each airport. We created a separate distribution for each airport.

7.1.2 Evaluation Metrics

The primary metric of interest for our evaluation is average per-aircraft delay for departures at each of the three airports modeled. We focus on departures because the multi-airport scheduler of PROCAST assumes gate holding as the primary means to manage departures; that is, the scheduler feeds back all the scheduled delay for each departure to its gate pushback time to avoid congestion and delays on the taxiways or at the runway queue. In our simulations, departure delay is composed of two components: *gate delay* and *taxi delay*. *Gate delay* is the difference between the departure's initial (e.g., airline-scheduled) gate pushback time (provided as input data to SOSS) and its actual gate pushback time in the SOSS simulation. *Taxi delay* is the difference between the departure's actual gate pushback-to-runway takeoff time and the unimpeded gate-runway transit time computed by SOSS. In the experiment platform, gate delay is assigned by the scheduler, and taxi delay is assigned by SOSS based on its CD&R logic. Gate delay is assigned by the scheduler to strategically meter and separate traffic as per the capacity constraints of the fixes and runways. Taxi delay is assigned by SOSS to tactically separate aircraft on the surface for safety (conflict detection and resolution).

For each airport we plot the average delay per aircraft for the baseline case of scheduling with deterministic taxi time models (i.e., the baseline scheduler), and for each of the scheduling cases listed in Table 13 which use the probabilistic taxi time models (i.e., the PROCAST probabilistic scheduler using different taxi time models). We expect that the use of probabilistic models with scheduling will improve the traffic performance achieved with scheduling in light of uncertainties or variability in traffic.

7.1.3 Experiment Matrix

Table 13 lists the six different scheduler configurations that we tested. For the PROCAST probabilistic scheduling, the number of futures generated and examined in each scheduling cycle can be varied. The baseline scheduler uses just one future, however predicted using deterministic models of transit time. The phrase "Sample, schedule, and down-select" in the table is a shorthand for our general probabilistic

scheduling approach using futures, as described in Section 4, comprising multi-airport scheduling for a single future and schedule specification from multiple scheduled futures.

For the special case of the Bayesian network model with one future, we did *not* sample from the predicted (posterior) distribution for each flight. Instead, we used the mean of this distribution directly as the prediction. Thus, scheduling in this case is the same as baseline scheduling, except that the estimate of departure taxi time, rather than being the deterministic unimpeded value, is obtained from our probabilistic Bayesian network model and is separately estimated for each flight.

Table 13. LEARN Phase II PROCAST Experiment Matrix

Departure Taxi Time Model	Number of Futures	Notes
Deterministic	1	PROCAST scheduling based on single future predicted from deterministic departure taxi time model No down-select due to deterministic transit time
Simple Probabilistic (Univariate Gaussian)	10	PROCAST scheduling based on multiple futures predicted from Gaussian distributed probabilistic departure taxi time model Sample, schedule, and down-select
Simple Probabilistic (Univariate Gaussian)	100	
Bayesian Network	1	PROCAST scheduling based on single future predicted from BN departure taxi time model No down-select due to using mean of posterior distribution of transit time
Bayesian Network	10	PROCAST scheduling based on single future predicted from BN departure taxi time model Sample, schedule, and down-select
Bayesian Network	100	

7.1.4 Traffic Scenarios

We tested each scheduler configuration with two two-hour traffic scenarios. The scenarios are derived from the actual schedules at the three New York metroplex airports on two specific days.

Table 14 lists the two dates and times uses, and the number of arrivals and departures at each airport in each scenario.

Table 14. Traffic Scenario Arrival and Departure Counts for PROCAST Experiments

Date, Time	EWR	JFK	LGA
7/25/12, 7-9 AM Local	Arrivals: 61 Departures: 68	Arrivals: 29 Departures: 64	Arrivals: 66 Departures:64
3/16/12, 8-10 AM Local	Arrivals: 73 Departures: 68	Arrivals: 77 Departures:64	Arrivals: 64 Departures:64

7.2 Results and Observations

We made runs of PROCAST using the different departure taxi time models as shown in Table 13 for each of the two traffic scenarios based on the same FAA sample days shown in Table 14. The average gate, taxi and total delay performances of departures from these runs are displayed in bar graphs in Sections 7.2.1 and 7.2.2. We have arranged these bar graphs to show the delays across all scheduling configurations for each scenario, and for each airport, separately (Figure 50, Figure 48, Figure 49, Figure 54, Figure 52, and Figure 53). We also include a comparable bar graph with data that is averaged across the three airports (Figure 51 and Figure 55).

7.2.1 Results for 7/25/12 Traffic Scenario

The departure delay performances obtained for the 7/25/2012 traffic scenario are presented for the individual airports JFK, LGA and EWR in Figure 48, Figure 49 and Figure 50, respectively, and for the three airports combined in Figure 51. Figure 48 below presents the departure delay results for JFK.

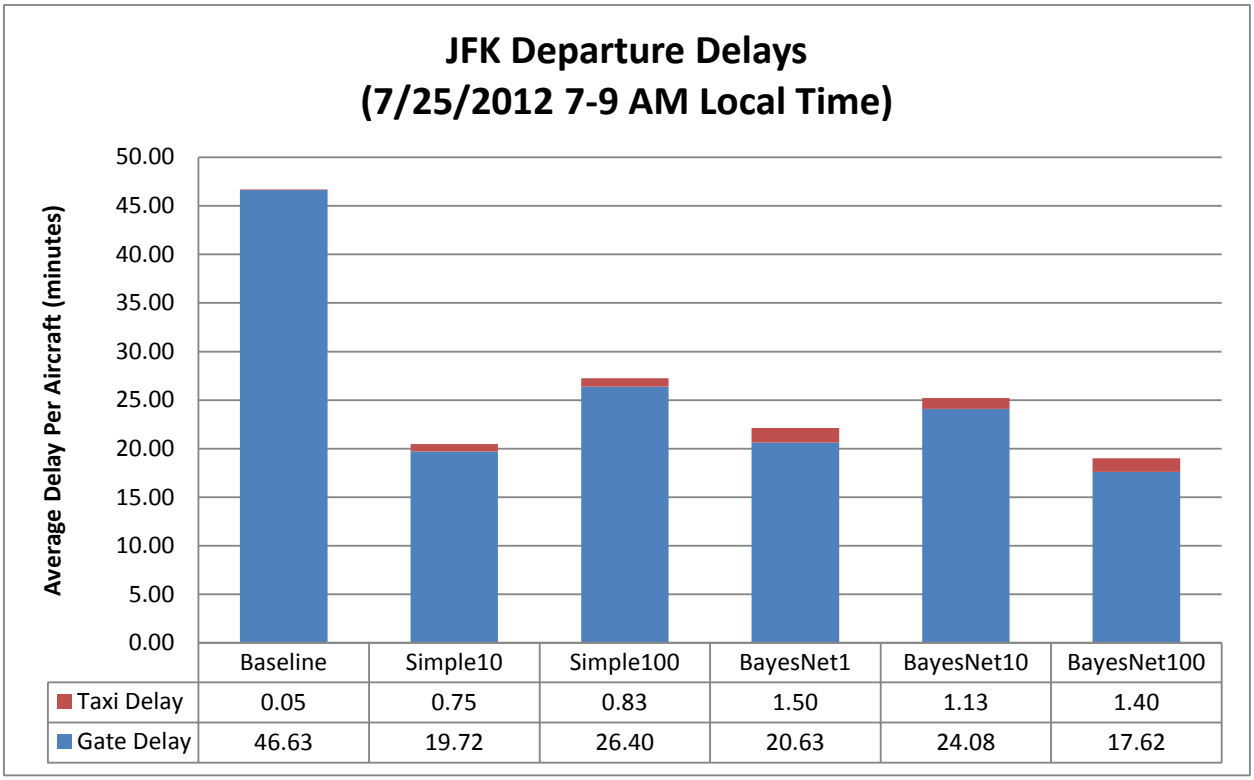


Figure 48. JFK Departure Delays for 7/25/12 Traffic Scenario

The results indicate that, in the baseline case of PROCAST using the deterministic departure taxi time model for traffic scheduling, the average total delay of the JFK departures is 46.6 minutes (Baseline), whereas in the cases of PROCAST using the different probabilistic departure taxi time models for traffic scheduling, the average total delays of the JFK departures are significantly less. The average total departure delays in the probabilistic cases range from 17.6 minutes with PROCAST scheduling based on 100 futures sampled from the BN model (BayesNet100), to 26.4 minutes with PROCAST scheduling based on 100 futures sampled from the simple univariate Gaussian model (Simple100). Thus, the reductions in average total delay for JFK departures realized by employing PROCAST traffic scheduling with the probabilistic taxi time models range from 20.2 minutes (Simple100) to 29.0 minutes (BayesNet100); these are significant reductions.

Comparison of the average taxi-out delays of PROCAST traffic scheduling between the baseline deterministic and the alternative probabilistic taxi time models indicates a slight trade-off in the probabilistic cases with the introduction of taxi-out delay. In the baseline case, the taxi delay is approximately zero, indicating the scheduled gate pushback times of the departures allow for essentially unimpeded transit between pushback and takeoff. In the probabilistic cases, the average total and gate delays are sharply reduced, however departures accrue some taxi-out delay as they encounter other aircraft on the surface. With the Simple10 model, average taxi-out delay is 0.75 minutes, and with the BayesNet1 model, average taxi-out delay is 1.50 minutes. Nevertheless, the average taxi-out delays of the departures are quite low. The negligible average taxi-out delay of departures in the baseline case would minimize JFK surface traffic congestion and the fuel burned by JFK departures while taxiing. However, the excessive average departure delay and the almost zero average taxi-out delay indicate the runways of JFK may be under-utilized by departures and average departure throughput of JFK may be lower than that of the probabilistic cases. In the probabilistic cases, the significant reduction in average total delay and the introduction of a small amount of taxi-out delay indicates PROCAST traffic scheduling using with the probabilistic taxi time models likely improves the utilization of the runways of the airport. Comparison of the departure throughput of JFK under the baseline and probabilistic cases would confirm this.

The differences in the departure delay performances of the different PROCAST probabilistic scheduling cases require deeper analysis and further investigation to understand and explain. Comparing the results of PROCAST traffic scheduling using the different probabilistic taxi time models, our hypothesis proposing improved traffic performance with more refined probability modeling and increased sampling was not clearly confirmed. The BayesNet100 condition of PROCAST scheduling using 100 futures sampled from the BN model supports the hypothesis, as it provides the lowest average departure delay of 17.6 minutes and the second-highest average taxi-out delay of 1.4 minutes. However, the Simple100 condition of scheduling using 100 futures sampled from the simple Gaussian model resulted in average gate, taxi and total delays for departures which were higher than those obtained from the Simple10 condition of scheduling using 10 futures sampled from the simple Gaussian model. Similarly, the BayesNet10 case of scheduling using 10 futures sampled from the BN model resulted in higher average total delay, although slightly less average taxi-out delay, than the BayesNet1 case of scheduling using 1 future sampled from the BN model. A deeper analysis and comparison of the characteristics of the departure schedules produced in each case, including the inter-aircraft spacing and transit times inherent in each schedule, and the quantity of conflicts resolved by the SOSS CD&R functionality, would help to understand the differences in the departure delay results obtained for the different probabilistic cases.

Figure 49 below presents the departure delay results for LGA for the 7/25/12 traffic scenario.

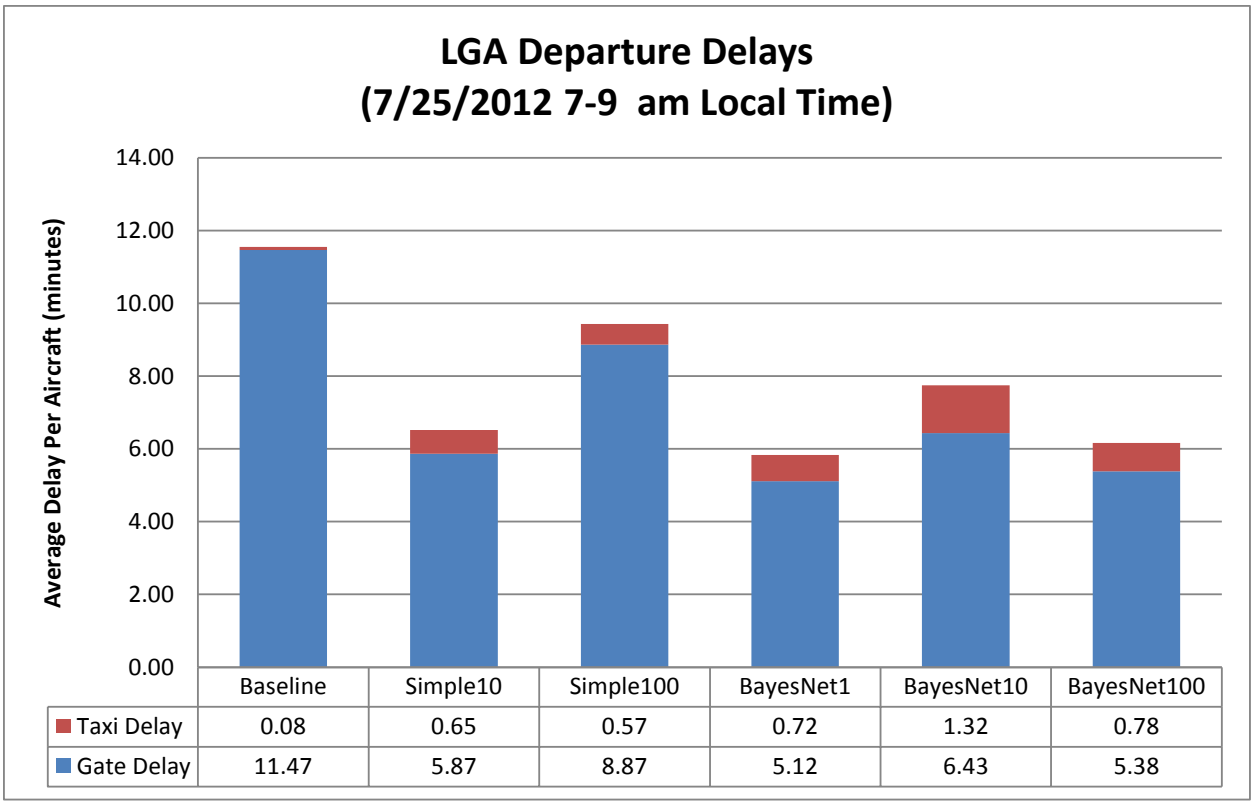


Figure 49. LGA Departure Delays for 7/25/12 Traffic Scenario

The results for LGA are similar to those for JFK. That is, the baseline case exhibits the highest average total delay and an average taxi-out delay of almost zero, and the probabilistic cases exhibit less average total delays and greater average taxi-out delay. However, for LGA, the difference in average total departure delay between the baseline case and probabilistic cases is less stark. Among the probabilistic cases for LGA, the trends in the total departure delays are similar to the JFK results, although for LGA the BayesNet1 case of scheduling with 1 future sampled from the BN model provides the lowest delay, slightly better than the BayesNet100 case of scheduling with 100 futures sampled from the BN model. Also, among all the taxi time model cases of LGA, the average taxi-out delay account for a higher percentage of the total departure delay than for the JFK cases.

Figure 50 below presents the departure delay results for EWR for the 7/25/12 traffic scenario.

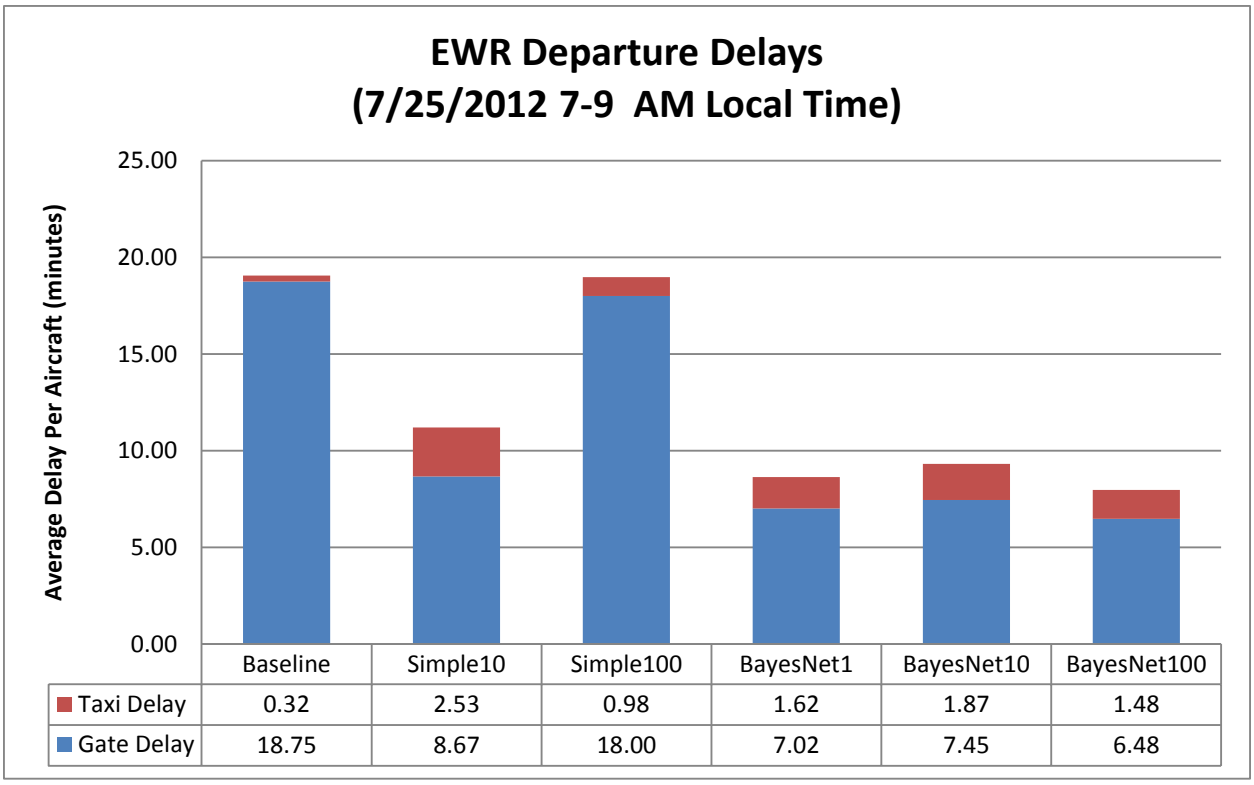


Figure 50. EWR Departure Delays for 7/25/12 Traffic Scenario

The results indicate that, for EWR, the BayesNet1, BayesNet10 and BayesNet100 cases of PROCAST scheduling using futures sampled from the BN model resulted in less average total departure delay than for the Baseline case of PROCAST scheduling using the deterministic taxi time model and the Simple10 and Simple100 cases of PROCAST scheduling using futures sampled from the simple Gaussian model. Thus, these results are in-line with our hypothesis of improved traffic performance with improved taxi time modeling and increased sampling. However, contradictory to our hypothesis, the average total delay of the Simple100 case exceeds that of the Simple10 case, and the average total delay of the BayesNet10 case exceeds that of the BayesNet1 case.

Figure 51 below presents the departure delay results for JFK, LGA and EWR combined for the 7/25/12 traffic scenario.

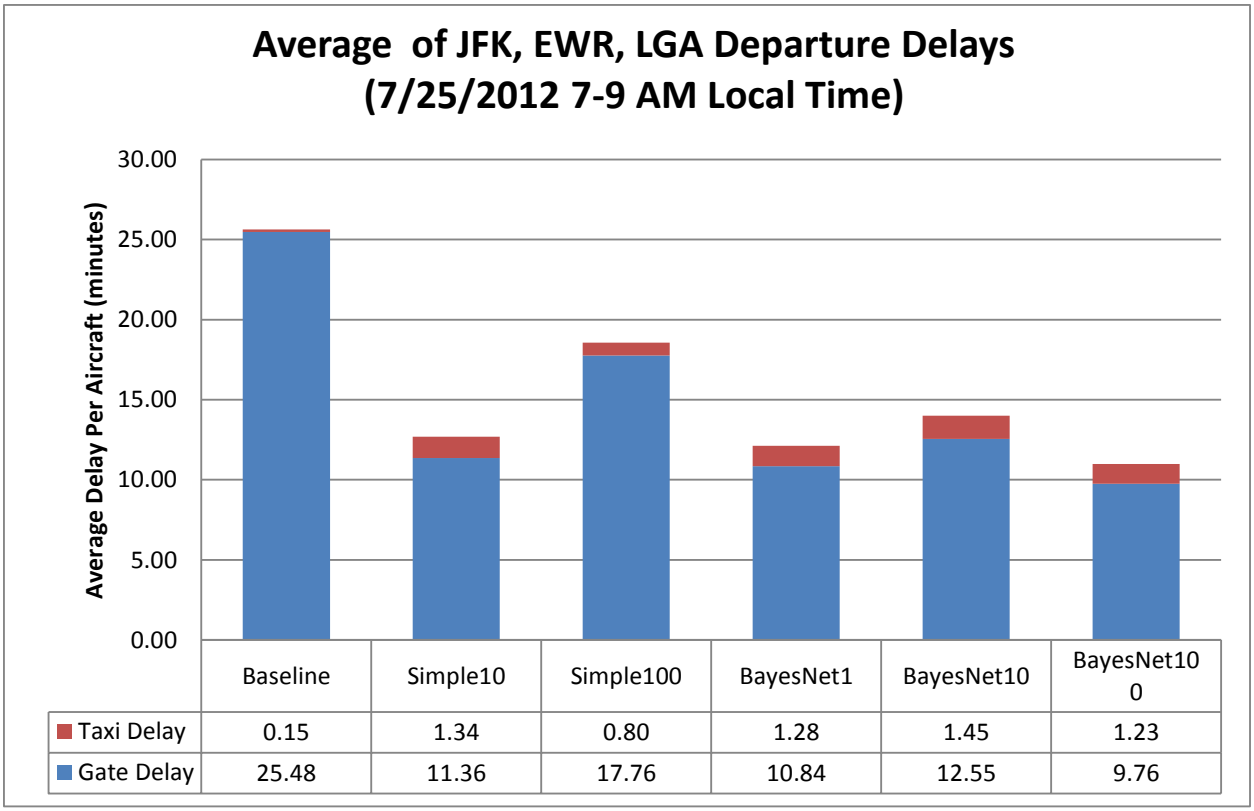


Figure 51. Average Departure Delays for EWR, JFK, and LGA for 7/25/12 Traffic Scenario

The results indicate that the Baseline case results in the highest departure delays with negligible taxi-out delays, and that BayesNet100 case of the PROCAST scheduling using 100 futures sampled from the BN model of taxi time resulted in the lowest average total delay and some marginal average taxi-out delay. The Simple100 case of PROCAST scheduling using 100 futures sampled from the simple Gaussian taxi time model produced the highest delays among the probabilistic scheduling cases. The BayesNet10 case resulted in higher average total and taxi-out delays than for the BayesNet1 case and the Simple10 case.

7.2.2 Results for 3/16/12 Traffic Scenario

The departure delay performances obtained for the 3/16/2012 traffic scenario are presented for the individual airports JFK, LGA and EWR in Figure 52, Figure 54 and Figure 53, respectively, and for the three airports combined in Figure 55. Figure 52 below presents the departure delay results for JFK in the 3/16/2012 scenario.

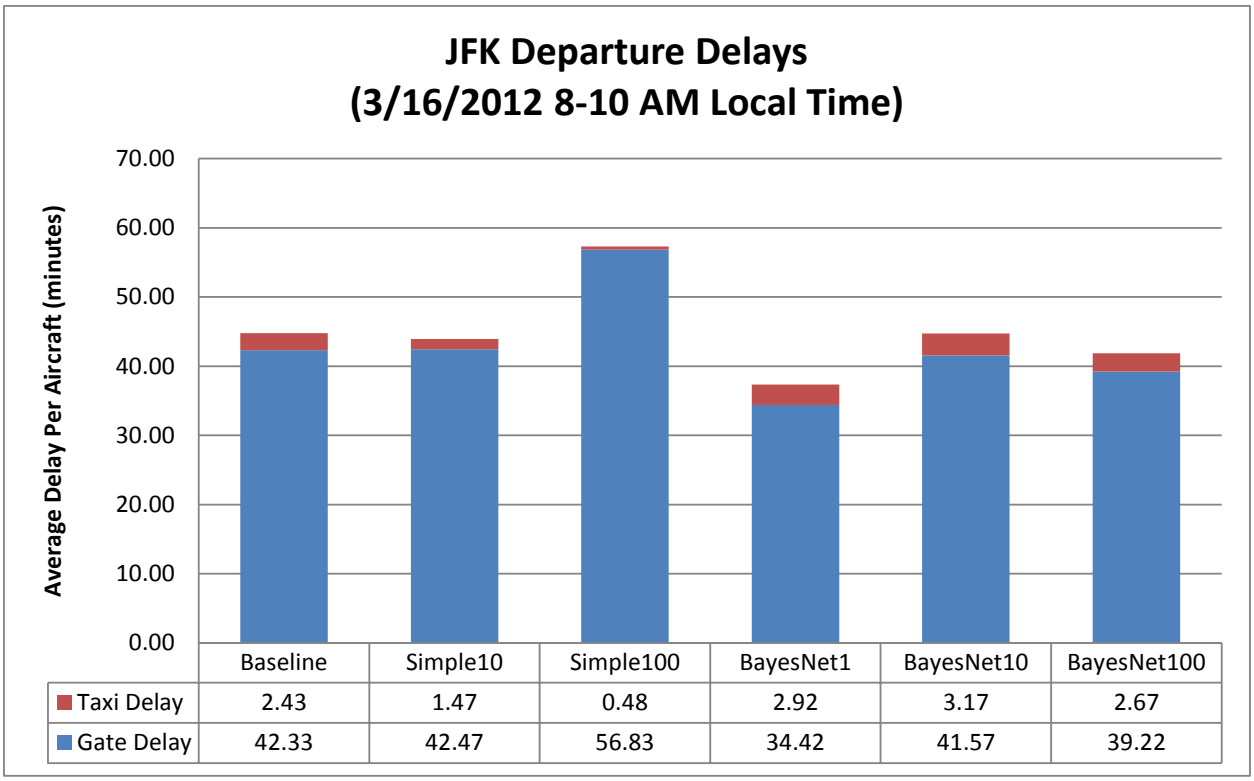


Figure 52. JFK Departure Delays for 3/16/2012 Traffic Scenario

The results indicate that the JFK departure delays realized with the probabilistic scheduling are much higher for this traffic scenario than those realized in the 7/25/2012 traffic scenario. In this traffic scenario, the average total departure delays for the Simple10 and Simple100 cases are 42.5 and 56.8 minutes, respectively, whereas for the 7/25/2012 scenario they are 19.7 and 26.4 minutes, respectively. Similarly, the average total departure delays for the BayesNet1, BayesNet10 and BayesNet100 cases are 34.4, 41.6 and 39.2 minutes, respectively, whereas for the 7/25/2012 scenario they are 20.6, 24.1 and 17.6 minutes, respectively. In addition, the total average departure delays for the Simple10 and Simple 100 cases exceed the delay of the Baseline case, a trend which is different from those observed in any of the airports for the 7/25/2012 traffic scenario.

Figure 53 below presents the departure delay results for LGA in the 3/16/2012 scenario.

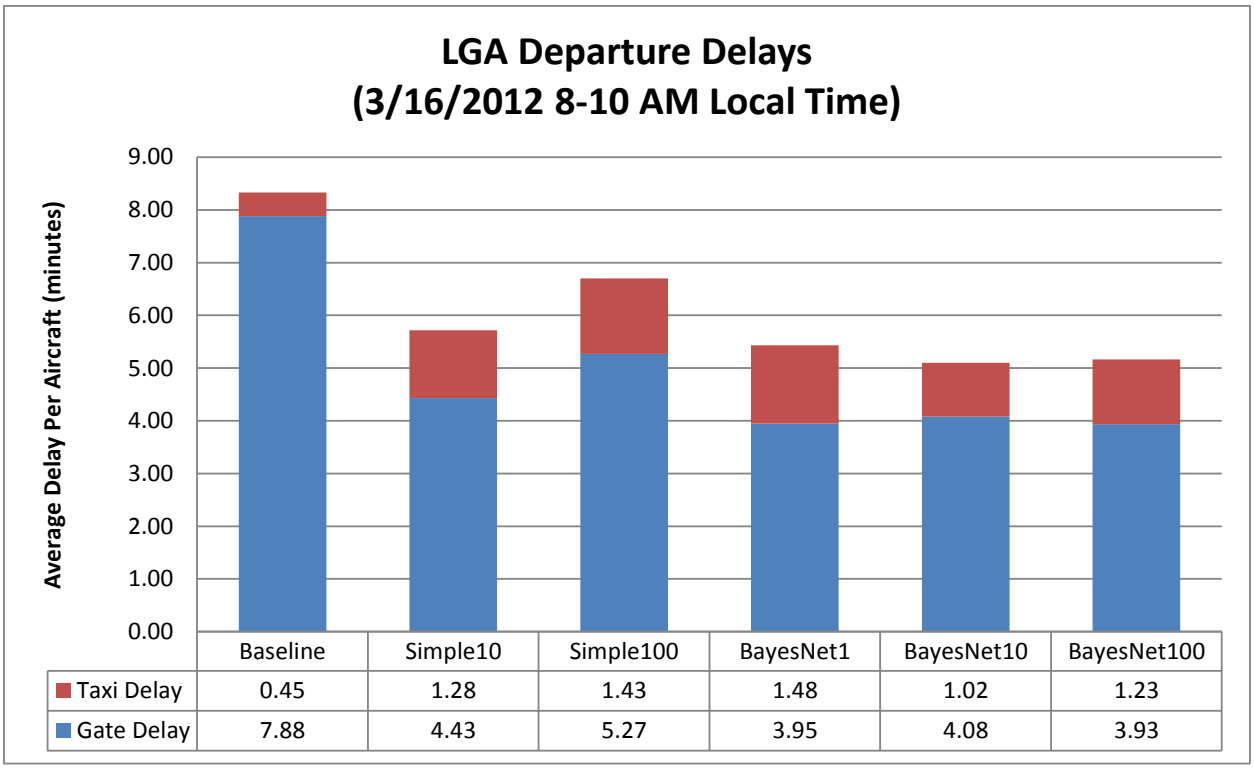


Figure 53. LGA Departure Delays for 3/16/2012 Traffic Scenario

In this traffic scenario for LGA, the magnitude of the average departure delay among the different taxi time model conditions, Baseline, Simple10, Simple 100, BayesNet1, BayesNet10 and BayesNet100, are somewhat lower than those in the 7/25/2012 case. In this traffic scenario, they range from 3.9 minutes for the BayesNet100 case to 5.3 minutes for the Simple10 case, and 7.9 minutes for the Baseline case. In the 7/25/2012 case, they range from 5.4 minutes for the BayesNet100 case to 8.9 minutes for the Simple100 case, and 11.5 minutes for the Baseline case. The average taxi-out times for departures obtained in this traffic scenario are also greater—and a greater portion of the total delay—than in the 7/25/2012 traffic scenario. Nevertheless, the trends in the delays obtained with PROCAST scheduling among the different departure taxi time models are similar to the 7/25/2012 results—with the exception of the average total delay in the BayesNet10 case being less than that of the BayesNet1 and BayesNet100 cases.

Figure 54 below presents the departure delay results obtained for EWR in the 3/16/2012 scenario.

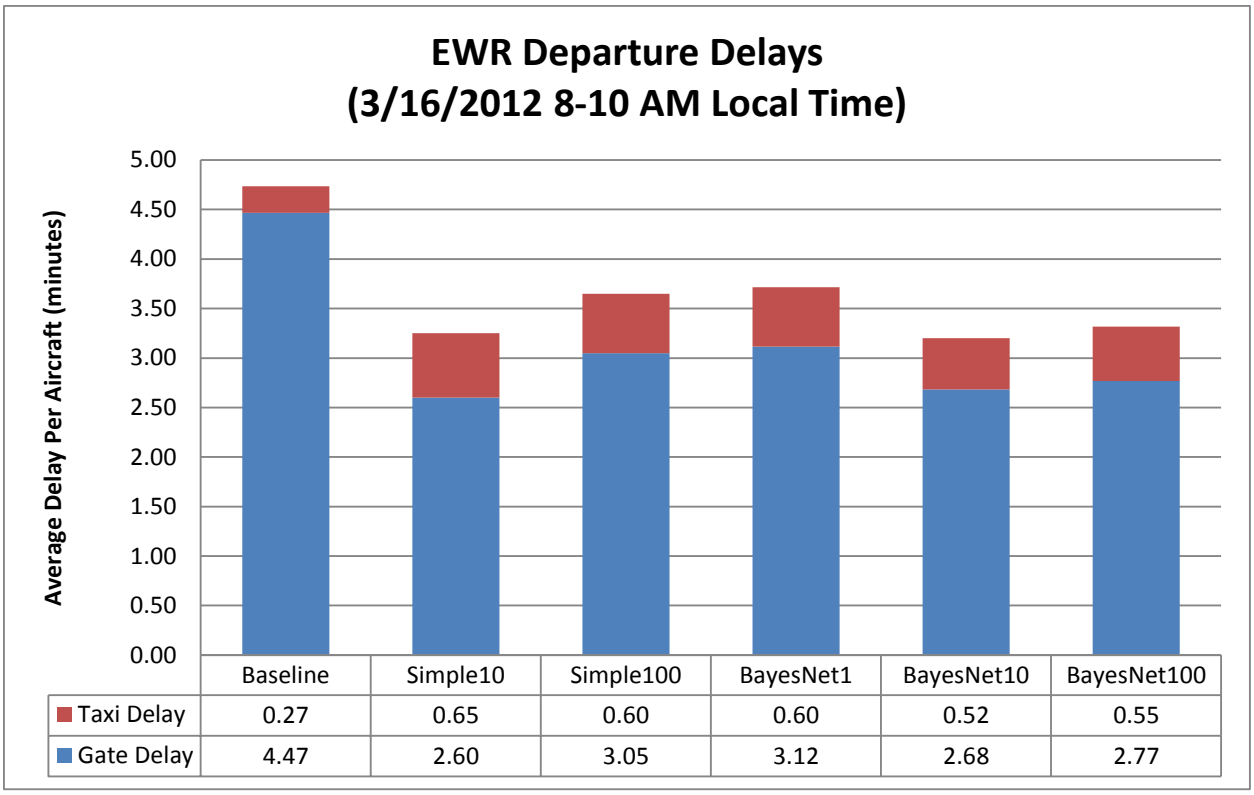


Figure 54. EWR Departure Delays for 3/16/2012 Traffic Scenario

In this traffic scenario for EWR, the magnitude of the average departure delay among the different taxi time model conditions, Baseline, Simple10, Simple 100, BayesNet1, BayesNet10 and BayesNet100, are also somewhat lower than those in the 7/25/2012 case. In this traffic scenario, they range from 2.6 minutes for the Simple10 case to 3.1 minutes for the Simple100 case, and 4.5 minutes for the Baseline case. In the 7/25/2012 case, they range from 6.5 minutes for the BayesNet100 case to 18.0 minutes for the Simple100 case, and 18.8 minutes for the Baseline case. The average taxi-out times for departures obtained in this traffic scenario are also lesser—however a greater portion of the total delay—than in the 7/25/2012 traffic scenario. Nevertheless, the trends in the delays obtained with PROCAST scheduling among the different departure taxi time models are similar to the 7/25/2012 results—with the exception of the average total delay in the BayesNet10 case being less than that of the BayesNet1 and BayesNet100 cases, and being greater than the Simple100 case.

Figure 51 below presents the departure delay results for JFK, LGA and EWR combined for the 7/25/12 traffic scenario.

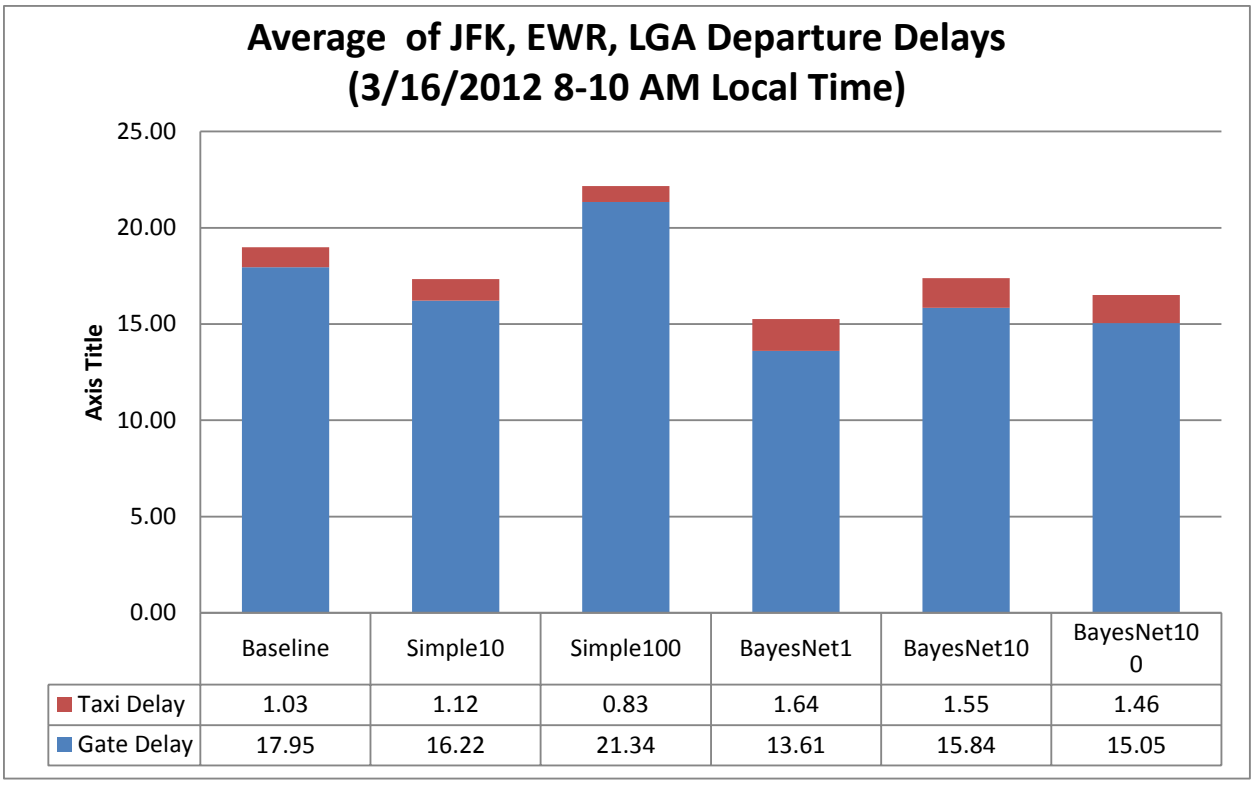


Figure 55. Average Departure Delays for EWR, JFK, and LGA for 3/16/2012 Traffic Scenario

The results indicate that the Simple100 case results in the highest departure delays and the smallest taxi-out delays among all the cases, and that BayesNet1 case resulted in the lowest average total delay and some marginal average taxi-out delay. The BayesNet10 case resulted in higher average total and taxi-out delays than for the BayesNet1 case.

7.2.3 Summary

In summary, w*ith one exception*, in each of the per-airport/per-scenario delay data, we observe the same pattern: The Baseline condition of PROCAST scheduling with deterministic taxi time models produces relatively large gate delays and very small taxi delays, and the other five instances of PROCAST scheduling using different probabilistic taxi time models produce larger taxi delays, and *usually* with significantly smaller gate delays. Figure 50 showing delays for EWR with the 7/25/12 scenario is a good illustration of this pattern. The total average delay per flight can be on the order of half as much as for the Baseline case, as in this example. In the Baseline case, most of the delay as per the schedule is allocated to the gate, not to taxi-out; taxi-out delay results from traffic interactions on the airport surface. The scheduling in the other probabilistic cases schedulers is producing more closely-packed schedules with smaller inter-aircraft spacing values, such that departures aircraft are delayed on the airport surface as they encounter congestion. In the Baseline case, the inter-flight spacing implicit in the schedule is sufficiently great to minimize occurrences of traffic congestion on the airport surface. The trade-off is that, in general, the resulting average flight delays are greater with the baseline scheduler than with the other schedulers. The ability of the PROCAST probabilistic scheduler to generate multiple, alternative runway sequences of flights, as well as the capability to accept a less accurate schedule, likely plays a role in this dramatic difference.

The outlier in this pattern is the “Simple100” scheduler configuration—PROCAST scheduling using 100 futures sampled from the simple Gaussian model of departure taxi time. The gate delays and total delays for this PROCAST configuration are less consistent, ranging from slightly more than the Simple10,

BayesNet1, BayesNet10 and BayesNet100 schedulers, as in most of the airports and traffic scenarios, to as much as or more than in the Baseline case (e.g. Figure 50, Figure 52). The Simple10 result is comparable to the BayesNet results. And, the Simple100 result varies considerably, but is also similar to the BayesNet results. The compact schedules of Simple10 can also be explained by effects (9) and (10).

In comparing the results (within this pattern) of the three Bayesian network cases, BayesNet1, BayesNet10 and BayesNet100 of PROCAST scheduling using with 1, 10, and 100 futures sampled from the Bayesian network model of taxi time, we typically see a small rise in gate delays in going from 1 to 10 futures, and a comparable small decrease when going from 10 futures to 100. But, these changes are small. It is difficult to discern a pattern in the small variations in taxi time in these three cases. In comparing them to the baseline, the fact that these BN runs are all similar to one another and dramatically different from the baseline, suggests that the important difference is in the mean of the predictions rather than the width. In particular, the BayesNet1 case is ***exactly*** the same as the baseline, except that it is using the posterior mean prediction rather than a deterministic unimpeded taxi time. Going to 10 or 100 futures with down-selection seems to make a relatively small difference. Indeed, it may be that the sampling and down-selection method is indirectly accessing the mean of the posterior distributions which is implicit (approximately) in the samples.

The *exception* to this pattern is the delay data for JFK airport with the 3/16/12 traffic scenario (Figure 52). In this data set, the baseline scheduler does not have particularly large total delays or gate delays, compared with other schedulers. And, the general pattern among the results for the non-baseline scheduler configurations is also different. In fact, the outsized gate delays and tiny taxi delays in this case are found in the Simple100 results. The scheduling methods produced similar delays, except for the Simple100 scheduler which had large gate delays and very small taxi delays. The best-performing scheduler in terms of overall delay, by a small amount, was the BayesNet1 scheduler.

Averages across all three airports, are less interesting because delays at the different airports are not really comparable, or in a similar range. Thus, the cross-airport average tends to wash out details, and give unfair weight to the airport with largest delays.

A deeper analysis and comparison of the characteristics of the departure schedule produced by each scheduler, including the inter-flight spacing and transit times inherent in each schedule, and the quantity of conflicts resolved by the SOSS CD&R functionality, might provide deeper insight into the results obtained from the simulations.

7.3 Investigation of Statistically Best Flight Release Times

Part of the Phase II research focused on improving PROCAST's method to choose the "statistically-best" set of control actions (i.e., flight release times). This section summarizes that research, which was not incorporated into PROCAST due to time constraints.

The BN models are used to generate a set of 100 futures. Each future contains a set of flight release times for the aircraft that are waiting at gates ready to depart.[3] We need to find a set of "statistically-best" flight release times that represent the delay control actions of scheduling aircraft to satisfy capacity constraints. For this study, we used seven different methods to 1) rank order the different futures, 2) choose the best future, and then 3) select the flight release times given by that future. In order to rank order the futures, each method calculated some function (to be specified below) of the delay of each flight within each future, and then calculated the sum of the resulting values to yield the measure for that future. The futures were ranked ordered in ascending order of this measure. We first describe two metrics that we used to compare the seven different methods, and then describe the seven different methods.

[3] For the remainder of this section, when we refer to flights, we are referring only to flights at the gate waiting to depart, which are the flights for which we need to figure out the best flight release times.

We used two metrics to evaluate the various methods for rank-ordering futures. The first metric is the sum of the actual delays[4] over all flights in the top ranked future. The second metric involves first calculating the mean absolute difference between all pairs of futures' flight delays for the first 2, 3… 100 futures, and then calculate the mean over all the numbers of futures. This metric is related to the spread in flight delays among the top 2, 3… 100 futures. The spread among the flight delays in the top few futures is a useful way to assess how "robust" the selected flight release times are---do the best futures tend to cluster in terms of flight delays?

The following seven methods were used to calculate the delay used for each flight as part of the above metrics. For all of these, we will refer to the mean delay for a flight across all the futures as the *mean* and the standard deviation of the delay for a flight across all the futures as the *standard deviation*.

1. Delay for each flight relative to the mean, but where delays less than the mean are set to zero rather than a negative number. The idea is to not excessively reward delays that are significantly less than the mean and possibly fail to penalize futures with some very high delay flights.
2. Absolute value of the difference between the delay for each flight and the mean. This operates on the hypothesis that having delays close to the mean are likely to be more robust, so this method strongly penalizes delays that are far from the mean on either side.
3. Absolute value of the flight's delay minus the mean and divided by the standard deviation. The idea is that flights with a higher spread in delays across the futures should not have their reductions or increases in delays count as much as flights with a lower spread.
4. If the flight delay is in the interval [mean – standard deviation, mean], then the delay is set to zero, otherwise the delay is the absolute value of the delay minus the mean. The idea here is to only reward significant reductions of flight delay below the mean.
5. Delay for each flight minus the mean divided by the standard deviation. If a flight's delay is less than zero, then set that flight's delay to zero.
6. Same as method 5, but where the score of the future is the mean of the square of the delays.
7. Same as method 5, but where the score of the future is the mean of the logs of the delays.

Table 15 gives the comparisons of all the methods in terms of the first metric, which is the sum of delays (in seconds) of all departing flights in the top-ranked future. If entry (i,j) is negative, then method i performed better than method j.

Table 15: Comparisons of Methods Based on Metric 1--- Total Delay Of All Flights In The Top-ranked Future.

METRIC 1							
	Method 1	**Method 2**	**Method 3**	**Method 4**	**Method 5**	**Method 6**	**Method 7**
Method 1	0	-2121.9	-2086	-1735	78.156	-186.28	-4095.4
Method 2	2121.9	0	35.867	386.92	2200	1935.6	-1973.5
Method 3	2086	-35.867	0	351.06	2164.2	1899.7	-2009.4
Method 4	1735	-386.92	-351.06	0	1813.1	1548.7	-2360.4
Method 5	-78.156	-2200	-2164.2	-1813.1	0	-264.43	-4173.5

[4] Not the function of the delays that are calculated as part of the seven methods.

Method 6	186.28	-1935.6	-1899.7	-1548.7	264.43	0	-3909.1
Method 7	4095.4	1973.5	2009.4	2360.4	4173.5	3909.1	0

Table 16 shows comparisons of all the methods in terms of the second metric, which is the mean absolute difference (in seconds) between all pairs of futures' flight delays, calculated for the first 2, 3, …, 100 futures. If entry (i,j) is negative, then method i has less spread in its flight delays than method j.

Table 16: Comparisons of Methods Based on Metric 2--- Mean Absolute Difference Between All Pairs of Futures' Flight Delays.

METRIC 2							
	Method 1	**Method 2**	**Method 3**	**Method 4**	**Method 5**	**Method 6**	**Method 7**
Method 1	0	-1.7699	-1.9283	-1.3707	0.021025	0.017103	-3.3079
Method 2	1.7699	0	-0.15837	0.39917	1.7909	1.787	-1.538
Method 3	1.9283	0.15837	0	0.55754	1.9493	1.9454	-1.3796
Method 4	1.3707	-0.39917	-0.55754	0	1.3917	1.3878	-1.9372
Method 5	-0.021025	-1.7909	-1.9493	-1.3917	0	-0.003922	-3.3289
Method 6	-0.017103	-1.787	-1.9454	-1.3878	0.003922	0	-3.325
Method 7	3.3079	1.538	1.3796	1.9372	3.3289	3.325	0

Table 17 gives the total delay of the all flights in the futures chosen by the new methods relative to the future chosen by the down selection method. The table compares the methods relative to down selection method on total delay (in seconds) of all flights in future chosen by the method relative to the future chosen by the down selection method. A negative value indicates that the new method has lower total delay than the down selection method

Table 17: Comparisons of the total delay of all the flights in the futures chosen by the new method relative to the futures chosen by the down selection method.

METRIC 3						
Method 1	**Method 2**	**Method 3**	**Method 4**	**Method 5**	**Method 6**	**Method 7**
-1850.4	271.46	235.6	-115.46	-1928.6	-1664.2	2245

Overall, Method 5 (highlighted in blue) has shown itself to be the best method, as it outperforms the down selection method and the other attempted methods. It remains for future work to investigate other possible methods. In particular, further analysis is needed to find better ways to assess the robustness of a

set of flight delays---how much can the system be perturbed and still allow the chosen set of flight delays, or values very close to them, to still be implemented?

8 Considerations and Future Research

This section presents considerations regarding probabilistic transit time and its incorporation into traffic scheduling, considerations in traffic scheduling, and considerations in future evaluations of PROCAST scheduling (Section 8.1). This section presents recommendations for future research, within the areas of multi-airport scheduling, Bayesian network modeling and application to traffic prediction and planning, and SOSS enhancements to improve modeling of arrival-departure interactions and multi-airport traffic (Section 8.2).

8.1 Considerations

Our experiments highlight a number of important considerations concerning PROCAST scheduler design and its evaluation which are described here. They include traffic control considerations, BN-based transit time prediction for traffic control, and scheduling.

8.1.1 Estimated Congestion Factors in Taxi Time Prediction

Predicting the 4D trajectories of departures from the BN models of taxi times introduced a couple key error sources in the PROCAST scheduling by 1) having to estimate the future congestion states of points on the aircraft surface, and 2) PROCAST scheduling inherently altering the congestion states of these points by altering the predicted 4D trajectories of aircraft to meet capacity limits and delay feedback requirements.

Regarding the first point, our Bayesian network models of departure taxi time accounted for traffic congestion at certain points on the airport surface as factors influencing the taxi time. For training of the BN models, such congestion states could be unambiguously determined from the known trajectories of the flights in the training data. However, to predict the taxi time of a departure for generating a future for scheduling required predicting or estimating the traffic congestion states at those points. This depended on the taxi times of the other current and pending aircraft and the pushback times of the pending aircraft, which are not known at the current time of prediction. We therefore used an approximation to the future trajectories of the flights to estimate the congestion state feature —an approximation that is *necessarily* cruder than the trajectory predictions that we have not yet made. To compute the values of the congestion point features of the BN model, we used the unimpeded taxi time of taxiing departures to estimate the future airport state. In addition, we assumed departures at the gate pushed back at the earliest time when they could push back, both for estimating node arrival times, and for estimating contextual traffic levels.

Regarding the second point, the PROCAST scheduling is inherently changing the congestion states of the congestion points by planning 4D trajectories of aircraft to comply with capacity constraints of the airport's runways and terminal airspace fixes. The transit time process of a departure flight after scheduling would likely be described by a different probability distribution than the posterior distribution used to predict the departure's transit time for scheduling. Developing an appropriate method, perhaps an iterative transit time estimation and scheduling approach, to account for the influence of scheduling in the initial transit time estimation, remains to be explored. Criteria to guarantee convergence for such an iterative approach would have to be specified.

Nevertheless, the Bayesian network prediction of the taxi time (the mean) has the potential to be more accurate than the deterministic unimpeded transit time in accounting for the particular traffic conditions at the time of the flight. While BN models may predict times of arrival to scheduling points which greater or lesser than a mean value, unimpeded transit times will tend to predict times of arrival to scheduling points which are too early, by not accounting for the impact of other traffic which can cause delays.

8.1.2 Traffic Scheduling

PROCAST scheduling with fixed, deterministic taxi time models will exhibit no permutations in the sequence of a given set of flights it schedules at the airport's runways. The combination of taxi-out time randomization and the down-selection process used in the PROCAST probabilistic scheduling admits more permutations of the sequence of the departing flights, and ***may*** be able to take advantage of this to make a more compact schedule. However, airport traffic performance resulting from the more compact schedule ***may*** exhibit greater variability in the face of taxi time fluctuations.

Probabilistic scheduling may be allowing us to choose slightly more inaccurate schedules relative to the actual conditions, and this may have the effect of making the scheduling problem dramatically easier to solve. We can think of this as an application of the Pareto principle [JMJ75], in which an 80% solution (a schedule that only approximately meets the constraints) is accomplished with 20% of the effort (imposing much smaller gate delays). The inaccuracy can be expected to appear as larger taxi delays.

8.1.3 Traffic Control Modeling Approximations

Schedule robustness versus throughput. Conflict situations among aircraft on the airport surface can result from the combination of 1) the dense traffic schedules observed to be generated by the PROCAST probabilistic scheduling and, 2) the uncertainties in the times of arrival of departures to the scheduling points due to taxi time variabilities. Such conflict situations are detected and resolved by airport traffic managers and controllers. Resolutions typically require holding one aircraft while letting the other pass, or rerouting aircraft along a different 3D route, either of which typically extends the taxi times of the aircraft. Our simulations did not capture or resolve conflicts between arrivals and departures, or between arrivals, at the airport runways. Therefore, our simulations may have generated artificially lower delay results with correspondingly higher capacity results than would otherwise be realized.

Schedule stability. Successive iterations of traffic scheduling will likely alter, say, the scheduled runway times of arrivals and departures and associated pushback times of departures. Sensitivity of controllers, pilots, passengers and airlines to changes in these scheduled times, and time windows for *freezing* scheduled times, is important to consider in implementing a scheduling system. In our simulations, only additional delays to the time of release for a flight from a node (such as the gate) were allowed; the time of release could not be hastened. If the flight if it subsequently received a second time of release was is earlier than the first one, that second time was ignored. This was so, even if the second time is still in the future. Thus, flights that had been scheduled (by being given a gate pushback time) could not be rescheduled to an earlier gate pushback, only to a later one.

8.2 Future Research

The two-year research effort produced a number of valuable PROCAST research products including SOSS airport terminal and surface models of JFK, EWR, and LGA, an integrated arrival and departure scheduling capability, and an evaluation framework for developing Bayesian Networks to probabilistically model and predict surface transit times (taxi times) for aircraft. There are a number of areas of interest that warrant future research including the following.

8.2.1 Multi-Airport Scheduling

Future work on multi-airport scheduling includes enhancing the algorithms to improve the efficiency and realism of planned traffic, investigating methods for incorporating uncertainty information into traffic scheduling. Regarding improving the scheduling algorithms, heuristics and optimization features could be added to increase throughput, optimize arrivals with departures on shared runways, and to ensure fairness in scheduling multi-airport traffic to shared resources. In addition, investigating schedule stability requirements for multi-airport arrival and departure scheduling, and the development of stateless schedulers that do not require persistence of scheduling data between successive iterations. In addition, enhancing the modeling with spatial freeze horizons and planning time horizons, and realistic details of

metering systems, would promote more immediate transition to an operational tool. Regarding methods to incorporate prediction of taxi time *distributions* into scheduling, previous research has explored the trade-off between throughput and controller workload in accounting for time of arrival uncertainty in scheduling. Exploring the trade-off between throughput and number of conflicts to specify inter-flight spacing buffers to account for the dynamically-changing transit time uncertainty is an area of future research; explicit incorporation of time of arrival uncertainty (derived from transit time uncertainty) in scheduling times of arrival is one approach. Lastly, further work is required to study alternative methods for evaluating futures and down-selecting. In particular, machine learning may enable the scheduler to understand and respond to the structure in the set of futures, such as clusters.

8.2.2 Bayesian Networks

Future work on the Bayesian networks includes evaluating alternative BN modeling and training methods, exploring real-time training of the BN models, and more comprehensively evaluating the appropriate implementation of BN models with scheduling. Regarding BN modeling and training methods, investigating Bayesian networks to use a hybrid of discrete and continuous variables (rather than the purely discrete or continuous implementations in this project) would support optimizing the modeling of individual categorical and continuous factors in the BN model. Evaluating other BN learning and prediction methods to model aircraft taxi time might identify modeling methods which are more efficient, robust and/or accurate for training and implementation. Regarding real-time training of BN models, we consider that the character of taxi time, the factors influencing it, and its dependence on those factors may change with various exogenous variables. Further research could explore the online update of trained models (using historical batch data) by surveillance and other data feedback from previous scheduling cycles. In addition, further research is needed to explore the holistic integration of BN taxi time modeling with traffic scheduling. Schedules are derived from predictions of taxi times, but taxi times (and their predictions) are affected by scheduling. It would be interesting to study a more holistic approach in which prediction and scheduling are combined, either directly, or through iteration.

Also, in contrast to optimizing the traffic schedule for every cycle, research could explore performing a global optimization for all (or a large number) scheduling cycles together. The objective will be to maximize the total discounted reward over all scheduling cycles. As the scheduling actions for one-cycle affects the scheduling decisions for the subsequent cycles, a global optimization approach is likely to perform better. We can formulate this optimal decision making problem as a Markov Decision Process (MDP) and solve it using Reinforcement Learning (RL) algorithms. Planning algorithms can be an alternative solution to the same problem.

8.2.3 SOSS Enhancements

To support future research, enhancements to the SOSS software would greatly improve our simulation test environment. One key area is the simulation and management of airborne aircraft. SOSS was designed as an exclusively surface simulation tool; therefore simulation of the entire TRACON environment requires revisiting the handling of airborne aircraft. While some simplified airborne functionality was added to SOSS under the Phase I work, this functionality is inadequate for the coupled airport interactions modeled in Phase II. SOSS functionality could be expanded to include the spacing and de-confliction of arrivals, and arrivals with departures, and for delaying arrivals in SOSS. To capture the coupled nature of multiple airports operating in the Metroplex, a separate airborne trajectory modeler component would be ideal. This component would handle airborne aircraft from the runways to the fixes, and would have a rudimentary ability to emulate approach control and vector, space, and delay aircraft. Ideally, some portion of delay could be handled in the TRACON so that not all delay would need to be fed back to the arrival fixes. The trajectory modeler would be able to space arrivals appropriately for the runways and insure departure fix metering was performed. Figure 56 shows a hypothetical architecture. The airborne simulator would operate as a separate process and would manage socket communications between the SOSS processes. The airborne simulator would take charge of aircraft once they became

airborne and transfer them to SOSS once they landed. The airborne simulator would manage the communications between the external scheduler. Prior modifications to individual SOSSs to enable airborne operations would be suspended. Time synchronization between the processes would now be critical, so SOSS would need to be modified to operate on a time mechanism governed by the airborne simulator.

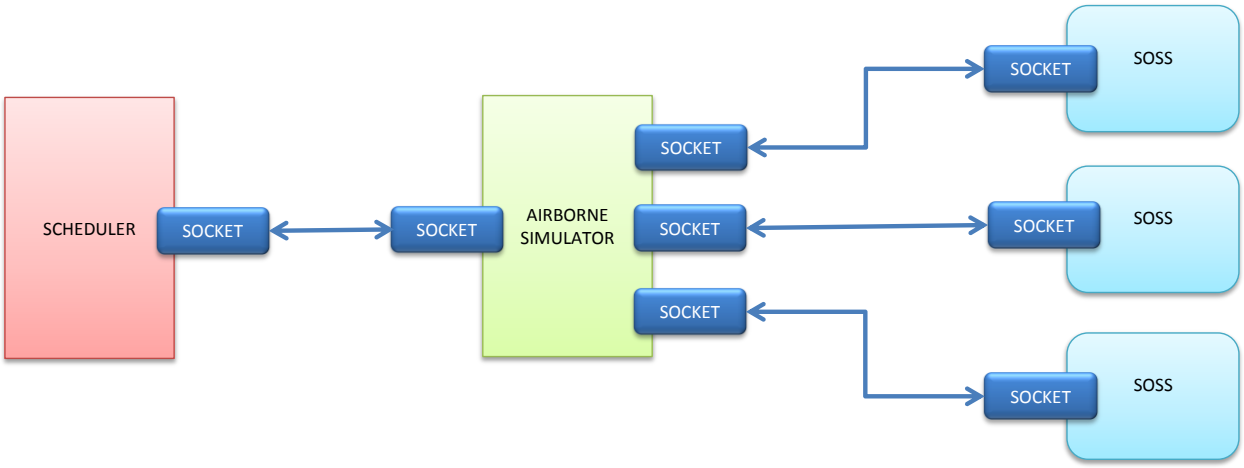


Figure 56. Hypothetical Architecture Involving an Airborne Simulator

9 Acknowledgments

We would like to thank NASA Aeronautics Research Institute (NARI) for sponsoring this research. Thanks to the NARI staff including Michael Dudley, Koushik Datta, Ricky Guest, Deborah Bazar, Cecelia Town, and Michael Tsairides for their support throughout the two year effort. Thanks to our NASA Technical Monitor Dr. Robert Windhorst for his feedback and guidance during our Phase II work. We also want to acknowledge the contributions of Ralph Tamburro (PANYNJ) and Bill Cotton (Cotton Aviation).

10 References

[ACES05] Airspace Concept Evaluation System (ACES), Contract Number: NNA05BE01C, CDRL 19 System/Subsystem Design Description (SSDD) / Software Design Document (SDD), 15 November 2005, Prepared for, NASA Ames Research Center, Moffett Field, CA 94035-1000.

[AS09] Aditya P. Saraf, David R. Schleicher , Katy Griffin, Peter Yu, Sylvester Ashok, Janae Bushman, and Victor Cheng, “Fast and Efficient Method for Generating Airport Models to Support National Airspace System Analyses,” AIAA Modeling and Simulation Technologies Conference, 10 - 13 August 2009, Chicago, Illinois, AIAA 2009-5914.

[AS14] Aditya Saraf, Kris Ramamoorthy, Steven Stroiney, Bruce Sawhill, Jim Herriot, “Robust, Integrated Arrival-Departure-Surface Scheduling based on Bayesian Networks,” 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), October 2014.

[AS15] Aditya Saraf, Valentino Felipe, Bruce Sawhill, “A Robust and Practical Decision Support Tool for Integrated Arrival-Departure-Surface Traffic Management,” 15th AIAA Aviation Technology, Integration, and Operations Conference, Dallas, TX, June 22-26, 2015.

[CA98] Carr, G., Erzberger, H., Neuman, F., “Airline Arrival Prioritization in Sequencing and Scheduling,” 2nd USA/Europe Air Traffic Management R&D Seminar, Orlando, FL, 1-4th December 1998.

[DR11] Daly, Rónán, Qiang Shen, and Stuart Aitken. "Learning Bayesian networks: approaches and issues." *The knowledge engineering review* 26.02 (2011): 99-157.

[FAADD] Future ATO Schedule and Data Analysis Tool (FASDAT) Data Dictionary V3.0

[HL15] Hanbong Lee, Waqar Malik, Bo Zhang, Balaji Nagarajan, Yoon Jung, "Taxi Time Prediction at Charlotte Airport using Fast-Time Simulation and Machine Learning Techniques," 15th AIAA Aviation Technology, Integration, and Operations Conference, Dallas, TX, June 22-26, 2015.

[JH13] Jeff Heaton, "Bayesian Networks for Predictive Modeling," Forecasting and Futurism, July 2013, Issue 7, Society of Actuaries.

[JMJ75] Joseph M. Juran, "The Non-Pareto Principle; Mea Culpa," Quality Progress, Vol. 8, No. 5 (May 1975), pp. 8-9.

[KD09] Koller, Daphne, and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

[LMI14] Analytical Identification and Ranking of Choke Points in the National Airspace System, Report NS304T2, Logistics Management Institute, Prepared for NASA Ames Research Center, September 2014.

[LMI15] Benefit-Cost Analysis of Deploying Air Traffic Management Technology Demonstration–2 to Address National Airspace System Choke Points, REPORT NS304T6, Logistics Management Institute, Prepared for NASA Ames Research Center, September 2015.

[ME14] Michelle Eshow, Max Lui and Shubha Ranjan, "Architecture and Capabilities of a Data Warehouse for ATM Research," 2014 IEEE/AIAA 33rd Digital Avionics Systems Conference (DASC), October 2014.

[MS10] Scutari, M. "Learning Bayesian Networks with the bnlearn R Package." *Journal of Statistical Software*, 35.3 (2010): 1-22.

[NST10] National Science and Technology Council, "National Aeronautics Research and Development Plan," February 2010.

[RN03] Neapolitan, R., "Learning Bayesian networks." Prentice Hall, 2003.

[RW12] Robert Windhorst, "Towards a Fast-time Simulation Analysis of Benefits of the Spot and Runway Departure Advisor," 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 17 - 19 September 2012, Indianapolis, Indiana.

[SD04] Signor, D., Davis, P., Lozito, S., Andre, A., Sweet, D., Wallace, E., "Efficient Air Traffic Scenario Generation", AIAA Paper 2004-6399, AIAA Aviation Technology, Integration and Operations (ATIO) Conference, September 20 -22, Chicago, IL, 2004.

[SS12] Systematic Assessment of Surface Optimization Functions—Methods and Metrics Report for the project entitled Adaptation of a Surface Traffic Management Tool to Multiple, Capacity-Constrained Airports, Document No.: 201115743-D04, Version: 1, June 15, 2012, NASA Contract: NNA11AC50C, Prepared for NASA Ames Research Center, Saab Sensis Corporation, 85 Collamer Crossings, East Syracuse, NY 13057.

[SS14] SOSS Input File Modifications to Support Terminal Airspace Modeling, Version 1, Jun 12, 2014, NASA Award Number NNX14AC74A, Prepared for NASA Ames Research Center, Saab Sensis Corporation, 85 Collamer Crossings, East Syracuse, NY 13057.

[SS15] Final Phase I Report for Project Titled "novel, Multidisciplinary Global Optimization under Uncertainty," Document No.: 840-035580, Version 1, April 13, 2015, NASA Award Number

NNX14AC74A, Prepared for NASA Ames Research Center, Saab Sensis Corporation, 85 Collamer Crossings, East Syracuse, NY 13057.

[TI06] Tsamardinos, Ioannis, Laura E. Brown, and Constantin F. Aliferis, "The max-min hill-climbing Bayesian network structure learning algorithm." Machine learning 65.1 (2006): 31-78.

[VC09] Victor H. L. Cheng , Gregory D. Sweriduk, Anthony Y. Seo, Wei-Man Lin, and Jack C. Yeh, Aditya Saraf, and David R. Schleicher, "Process and Software Tools for Generating Airport Models to Support Surface Operation Analyses," AIAA Modeling and Simulation Technologies Conference, 10 - 13 August 2009, Chicago, Illinois, AIAA 2009-5915.

[W00] Wong, G. L., "The dynamic planner: The sequencer, scheduler, and runway allocator for air traffic control automation" NASA/TM-2000-209586, April 2000

11 Appendix A: SOSS Adaptation Data

As in Phase I, we simulate the airport surface and terminal airspace traffic in order to evaluate the effectiveness of the PROCAST solution architecture using BNs to model taxi-times. We use a NASA airport surface traffic simulation platform called Surface Operations Simulator and Scheduler (SOSS) for this purpose. SOSS is a fast-time simulation platform used to simulate airport surface operations and support rapid prototyping of surface scheduling algorithms. SOSS is designed to be used in conjunction with external scheduling components (e.g., PROCAST). When integrated with external scheduler(s), it is SOSS's job to move aircraft on the surface according to the recommended schedule, and monitor and resolve separation violations and scheduling conformance. SOSS uses an underlying link-node airport model representing gates, ramps, spots, taxiways, crossings, and runways. Flight surface routes in SOSS are defined as ordered lists of nodes through the node/link network. SOSS uses a dynamic model to simulate the motion of aircraft along the airport surface routes. SOSS also has a tactical flight separation model which emulates tactical actions that pilots and controllers take to maintain safe separation between aircraft. Separation is handled differently for flights using the runway than for flights taxiing through the ramp, taxi, and queuing areas. SOSS requires a simulation traffic scenario consisting of scheduled flight times, routes, and runway configurations to conduct simulations. For detailed information about SOSS simulation capabilities, SOSS architecture, etc., see [RW12]. This appendix provides additional details on the runway configurations and adaptation data used to model JFK, EWR, and LGA airports in the Phase II effort.

11.1 JFK Runway Configurations and Surface Arrival and Departure Routes

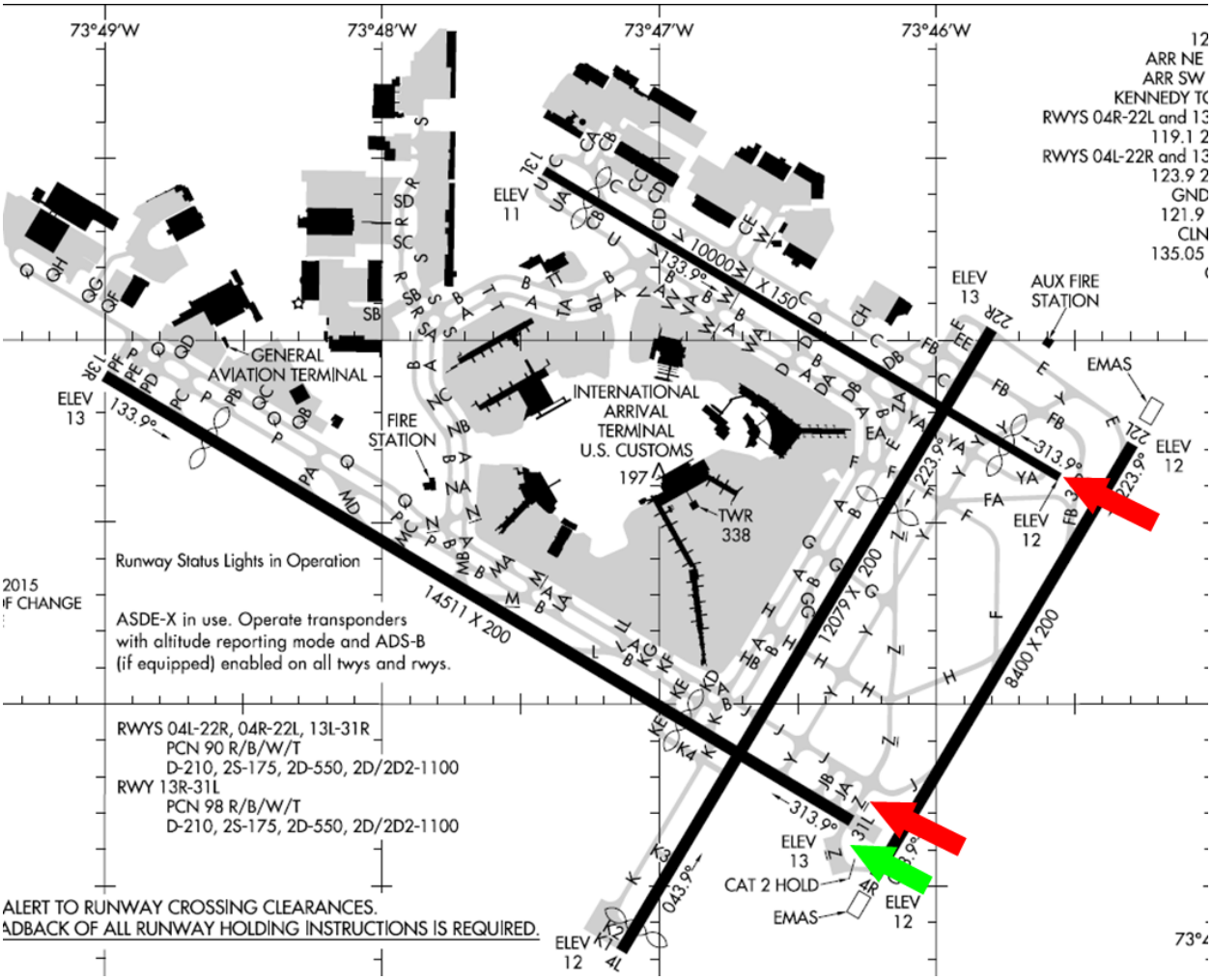


Figure 57. JFK Phase I and Phase II Runway Configuration Showing Arrivals in Red and Departures in Green (31L, 31R|31L)

11.2 Surface Arrival and Departure Routes

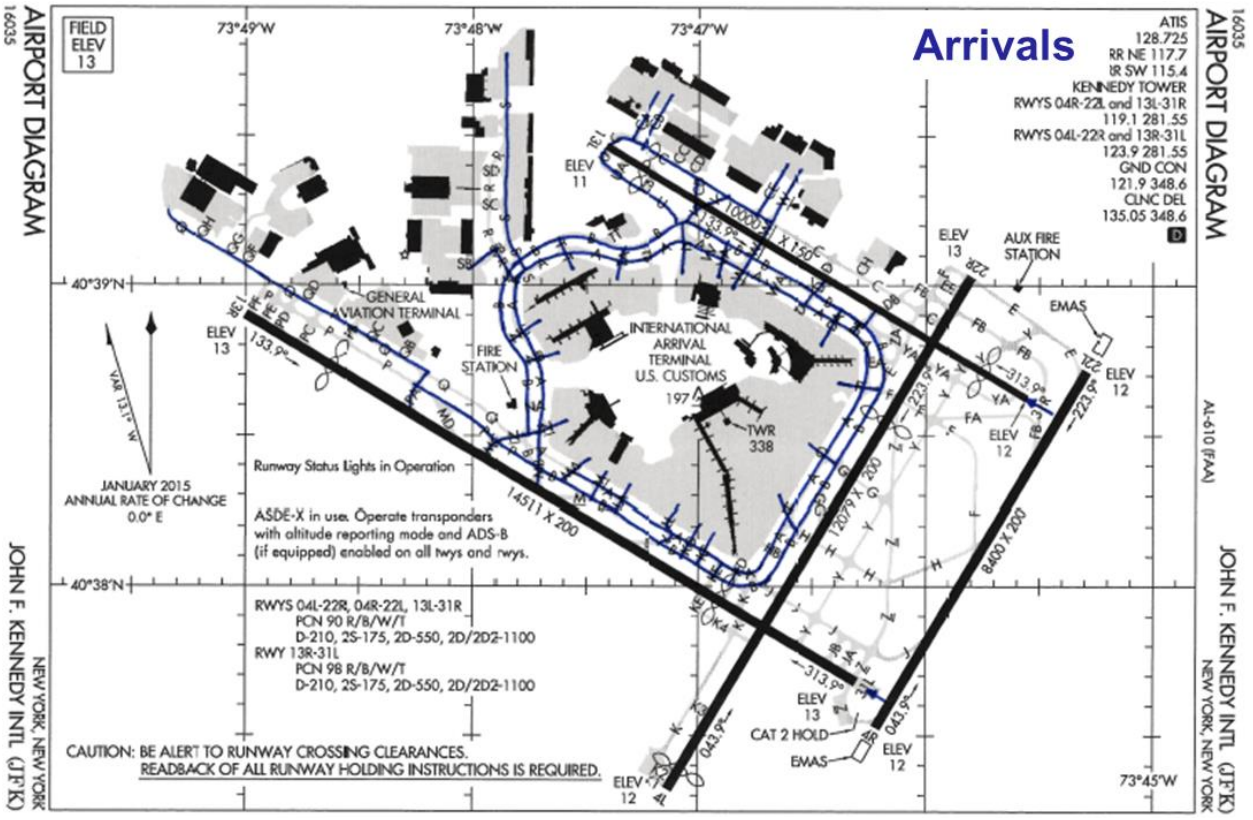


Figure 58. Illustration of JFK Surface Arrival Routes

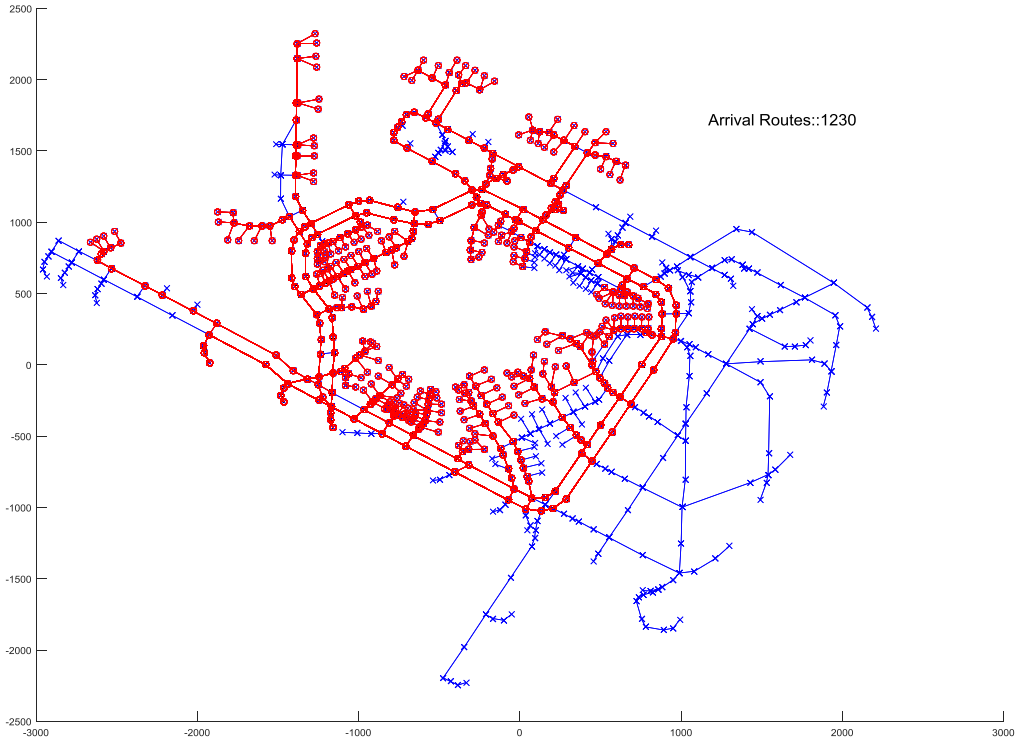


Figure 59. JFK Surface Arrival Routes in SOSS

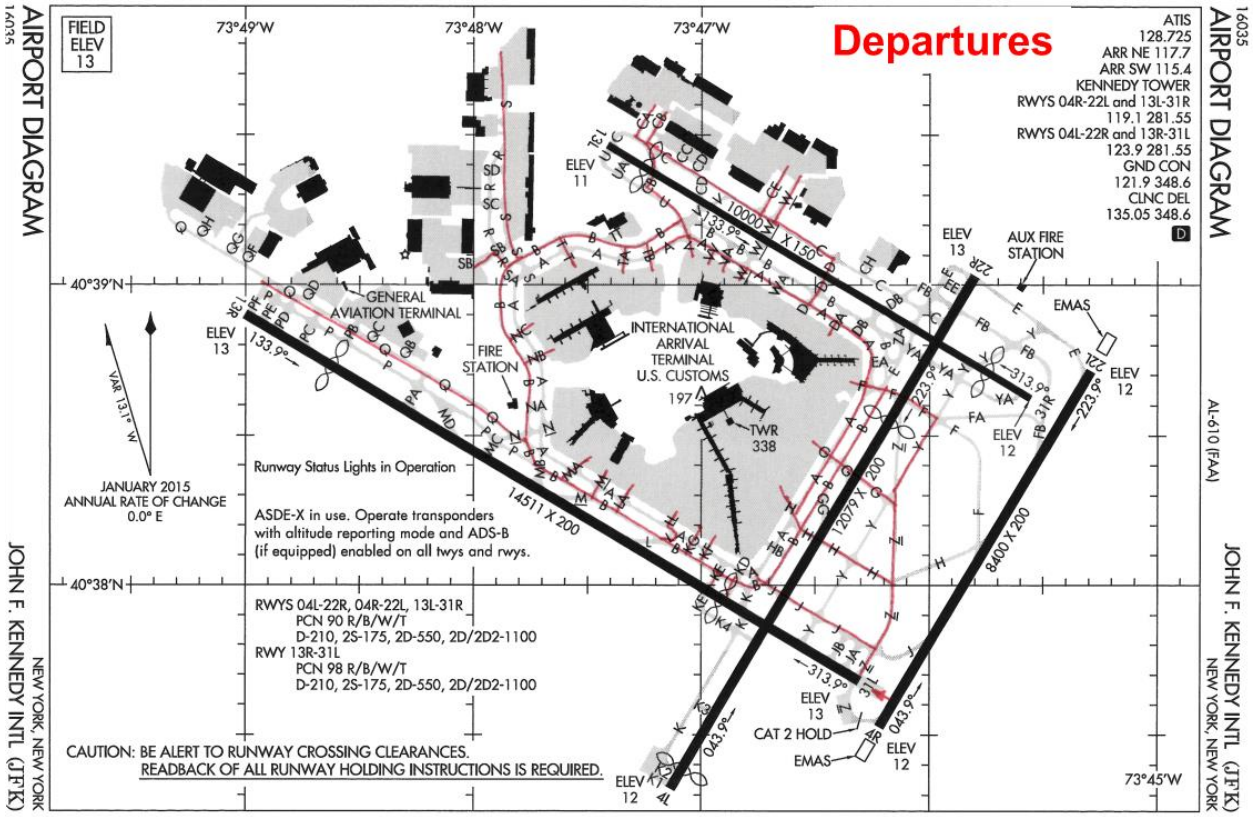


Figure 60. Illustration of JFK Surface Departure Routes

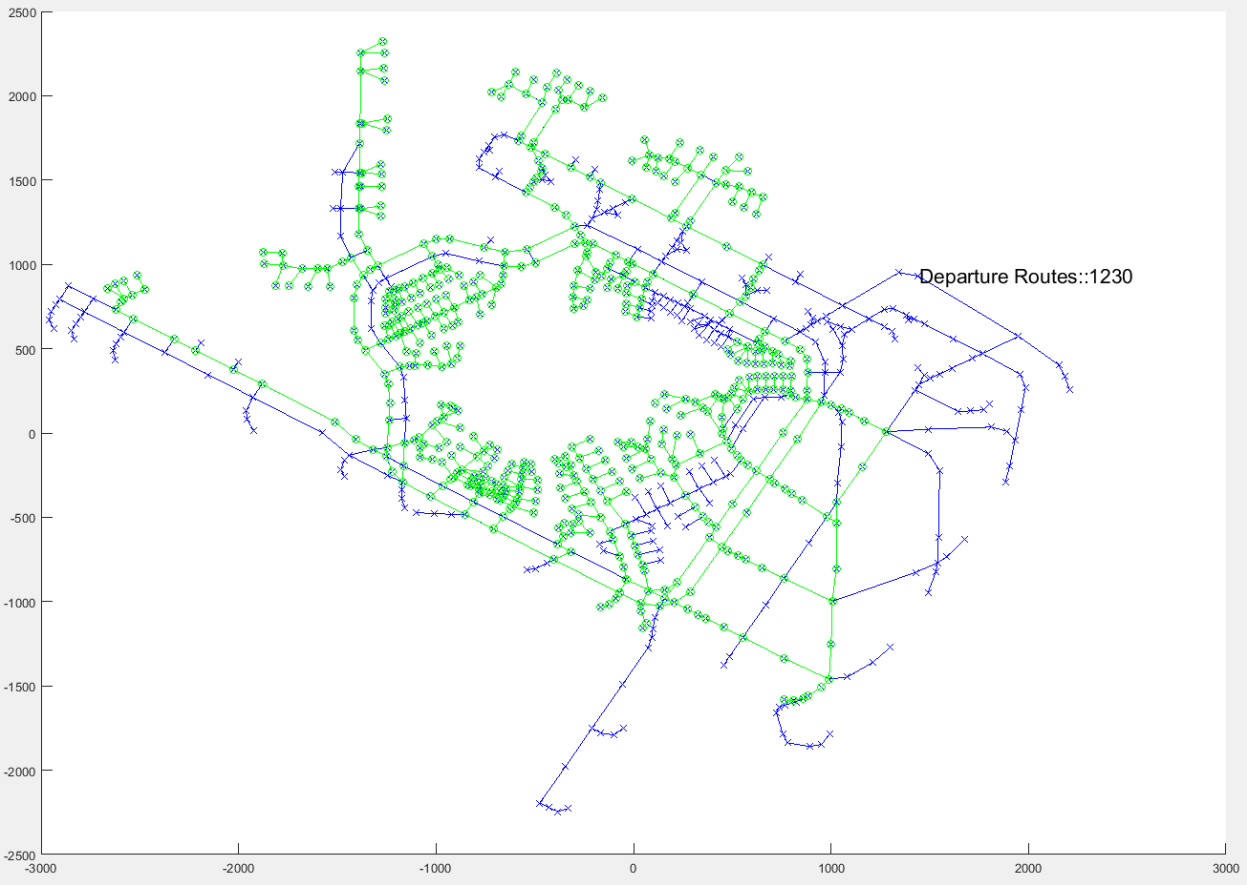


Figure 61. JFK Surface Departure Routes in SOSS

11.3 EWR Runway Configuration and Surface Arrival and Departure Routes

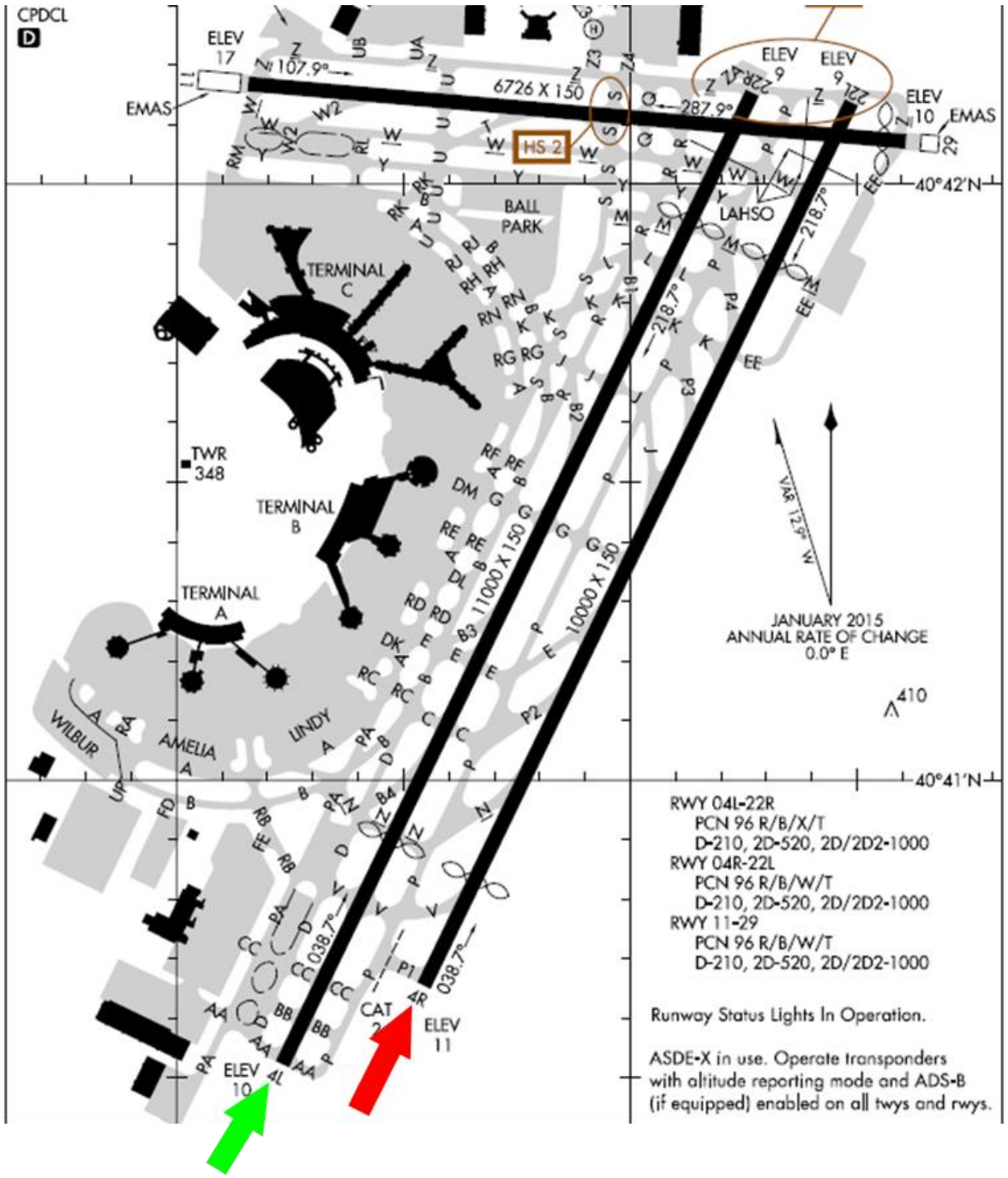


Figure 62. EWR Phase II Runway Configuration Showing Arrivals in Red and Departures in Green (04R|04L)

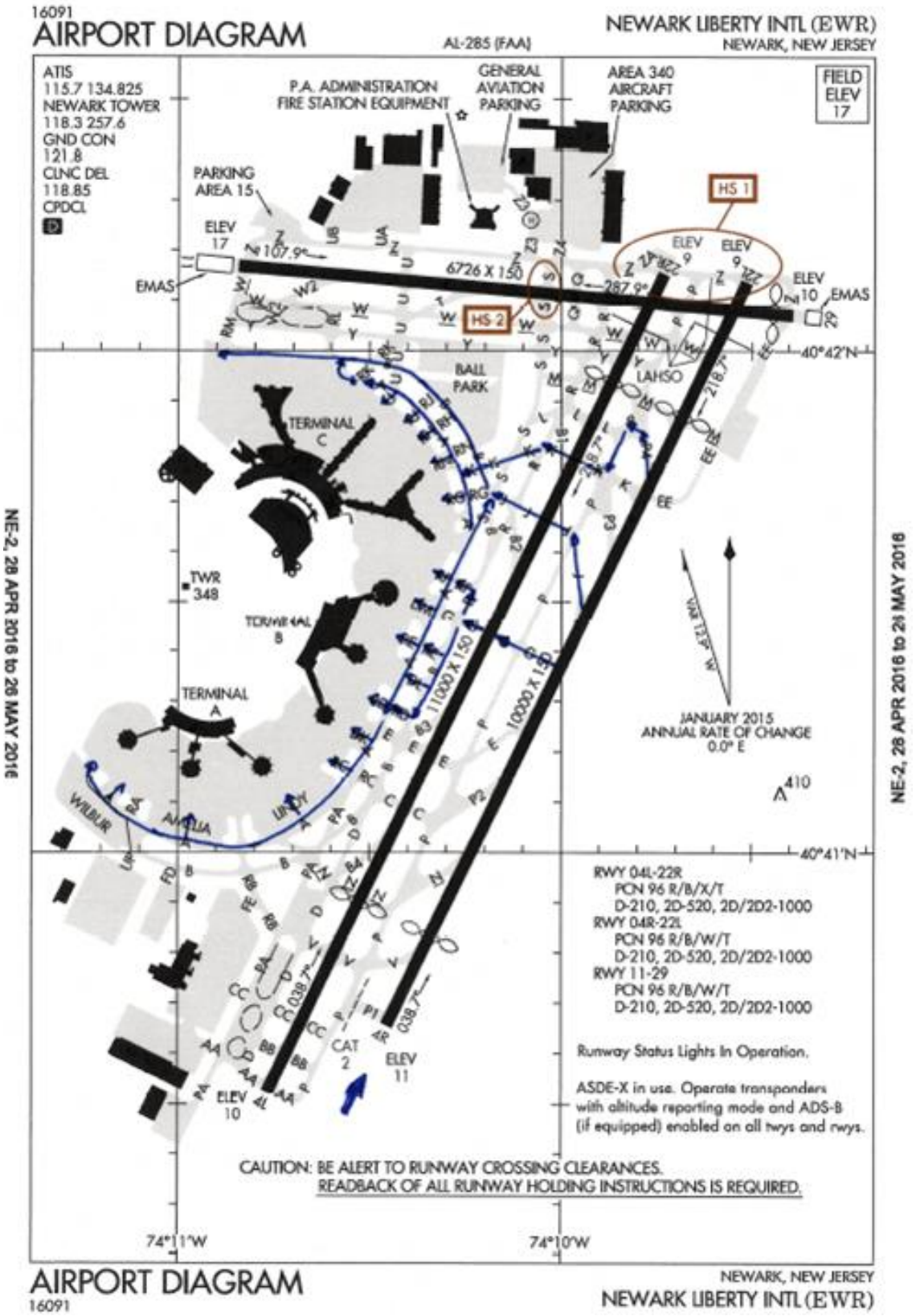


Figure 63. Illustration of EWR Surface Arrival Routes

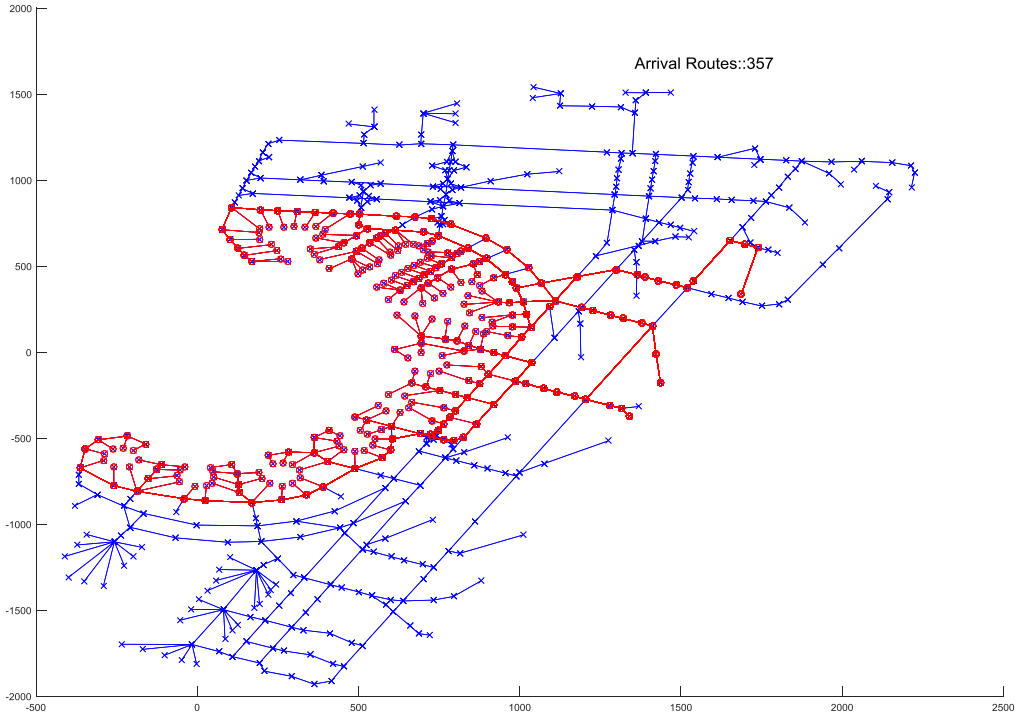


Figure 64. EWR Surface Arrival Routes in SOSS

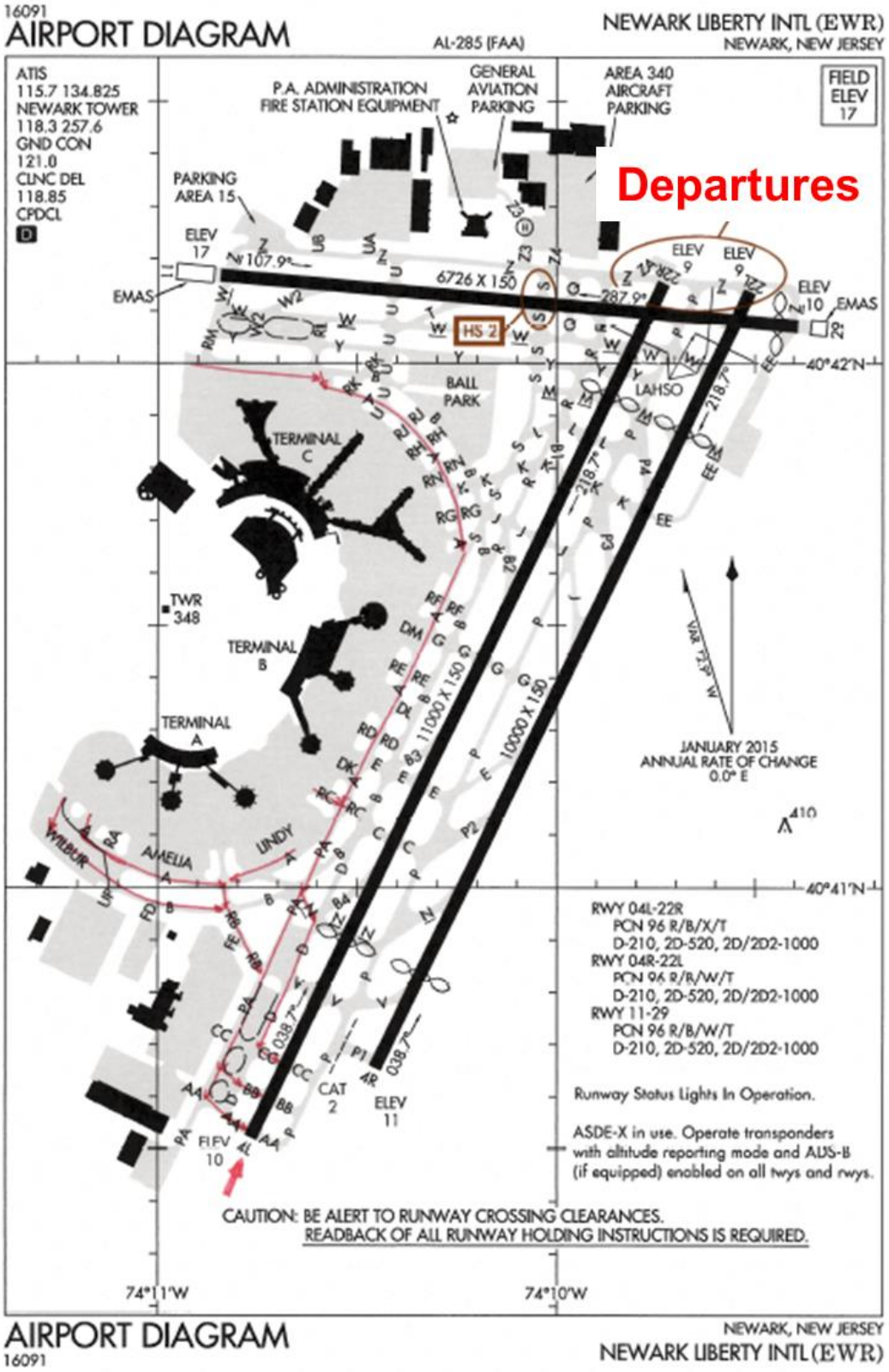


Figure 65. Illustration of EWR Surface Departure Routes

11.3.1 SME Review of EWR Surface Departure Routes

In order to better understand real-world influences that affect taxi-time for the surface routes modeled at EWR, we reviewed the illustration of the surface departure routes with our N90 SME's Ralph Tamburro and Bill Cotton. They offered the following comments on the taxi departure routes depicted in the illustration:

- EWR tends to use Bravo for Runway 4L departures.
- Terminal C departures enter taxiway Alpha at Romeo
- 4's are not a bad configuration because Terminal C is busiest
- The interactions typically occur near kilo (arrival/departure crossings) and Juliet
- Kilo would likely get the most runway crossings by arrival aircraft
- During a heavy departure push, bunch several arrivals to get them across in a group (or a burst)
- FedEx runway crossings are primarily on the south side of airport
- United feeder (express) flights are in terminal A
- Terminal B ramp operations are run by PANYNJ – primarily international, DAL is also terminal B – lower number of ops compared with Terminal C
- Can use taxiways Alpha and Bravo together for sequencing departures to meet restrictions
- Typical weather day do not put departures over the same fix back to back
- Sometimes difficult at EWR due to concentrated demand over certain fixes (e.g. International push during evening) Internationals use a different sector in the departure facility, but a common departure fix.
- Juliet and Kilo would likely be primary intersection/constraint points for this configuration.
- May need to cross arrivals to prevent an arrival queue on Papa.
- Arrivals come off 4R to Papa and then crossing at Echo and Gulf simultaneously.
- UAL works to manage surface traffic with EWR tower (nothing official/software), but they work collaboratively with ATC to self-manage traffic.

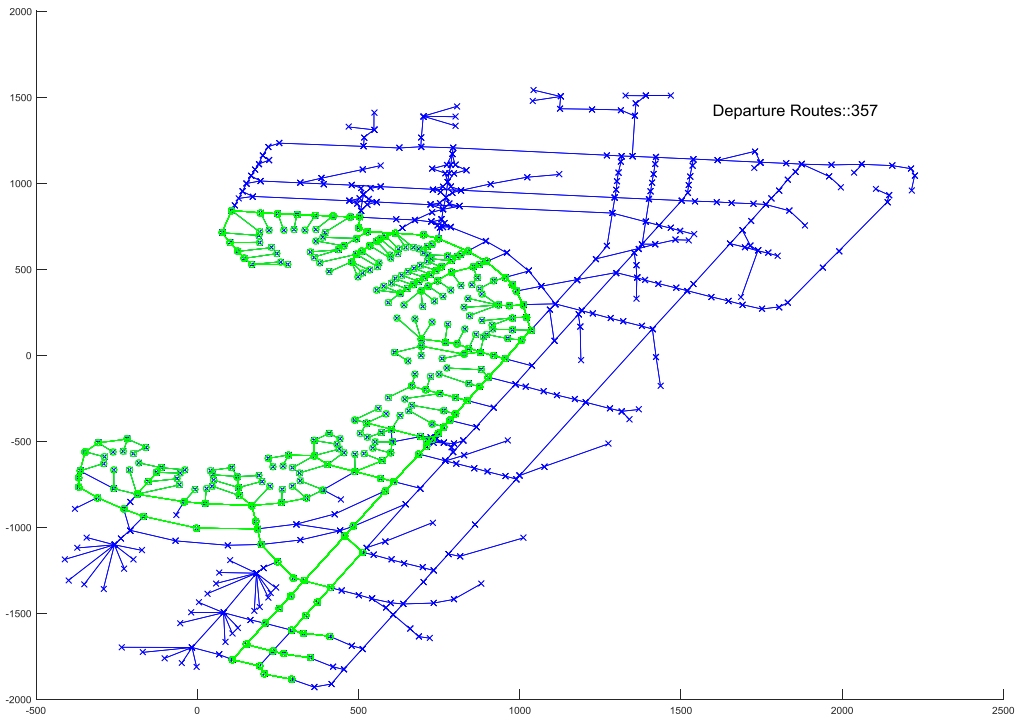


Figure 66. EWR Surface Departure Routes in SOSS

11.4 LGA Runway Configuration and Surface Arrival and Departure Routes

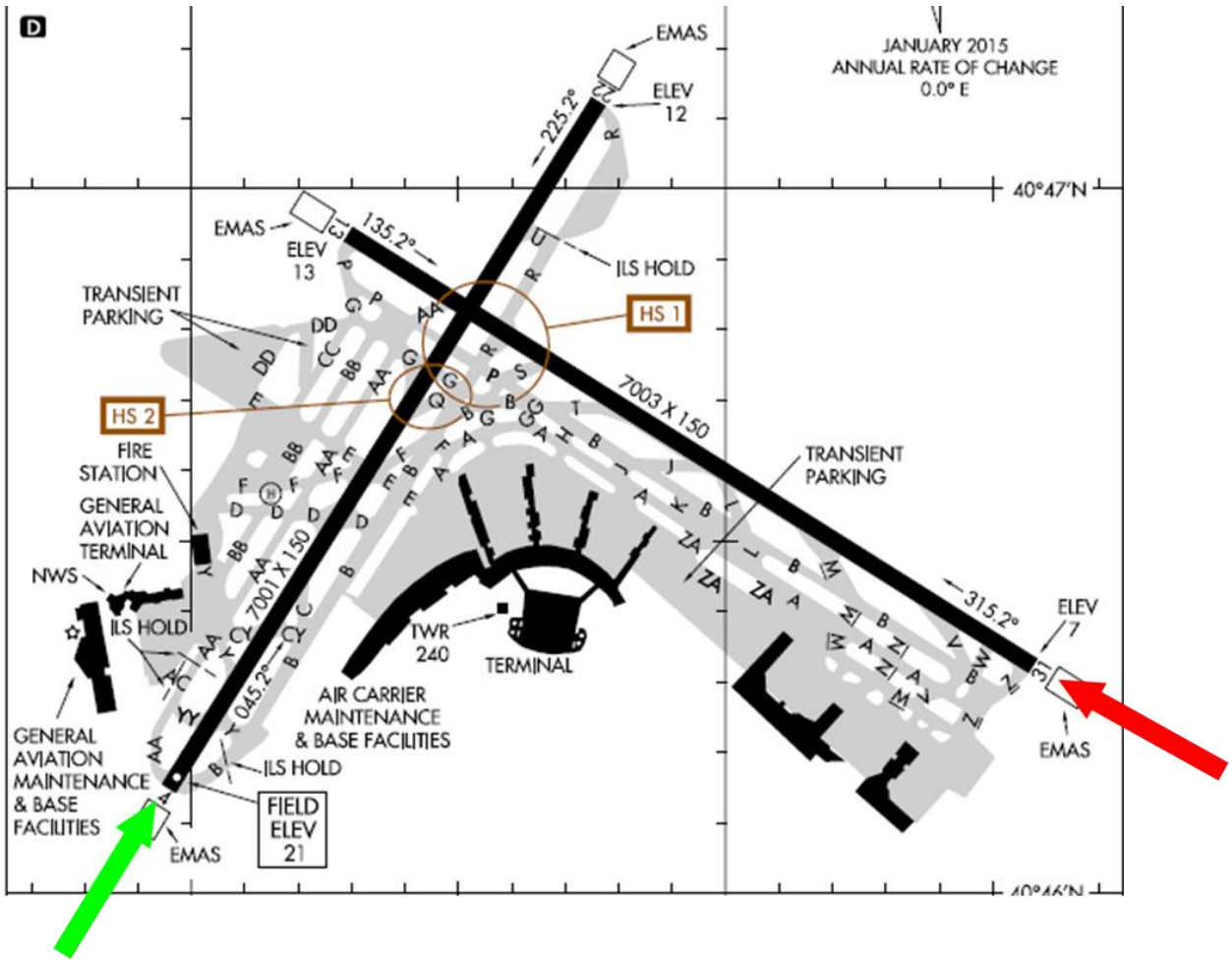


Figure 67. LGA Phase II Runway Configuration Showing Arrivals in Red and Departures in Green (31|04)

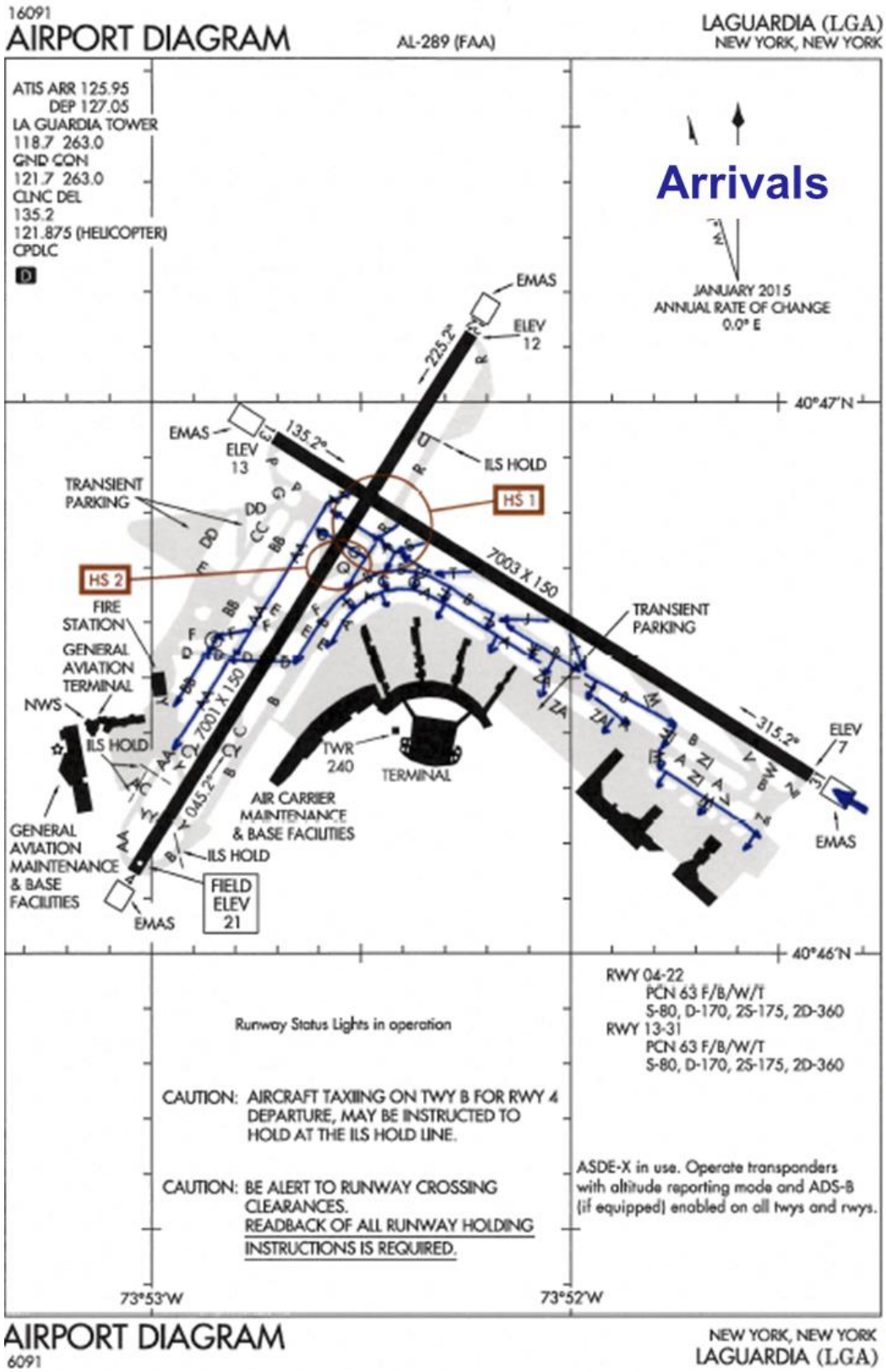


Figure 68. Illustration of LGA Surface Arrival Routes

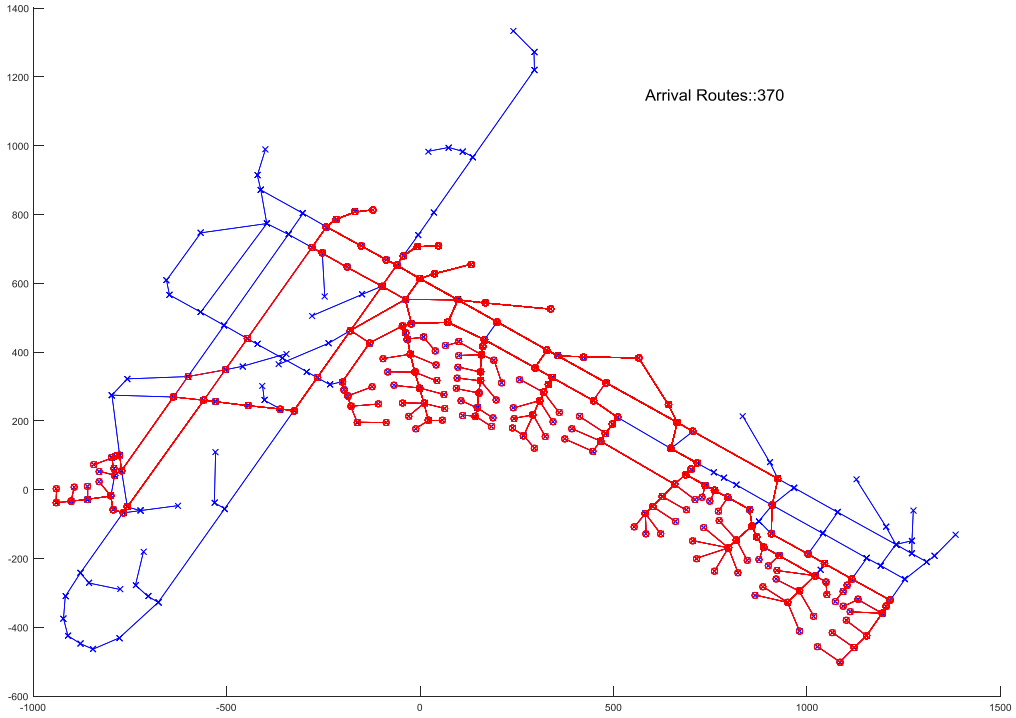


Figure 69. LGA Surface Arrival Routes in SOSS

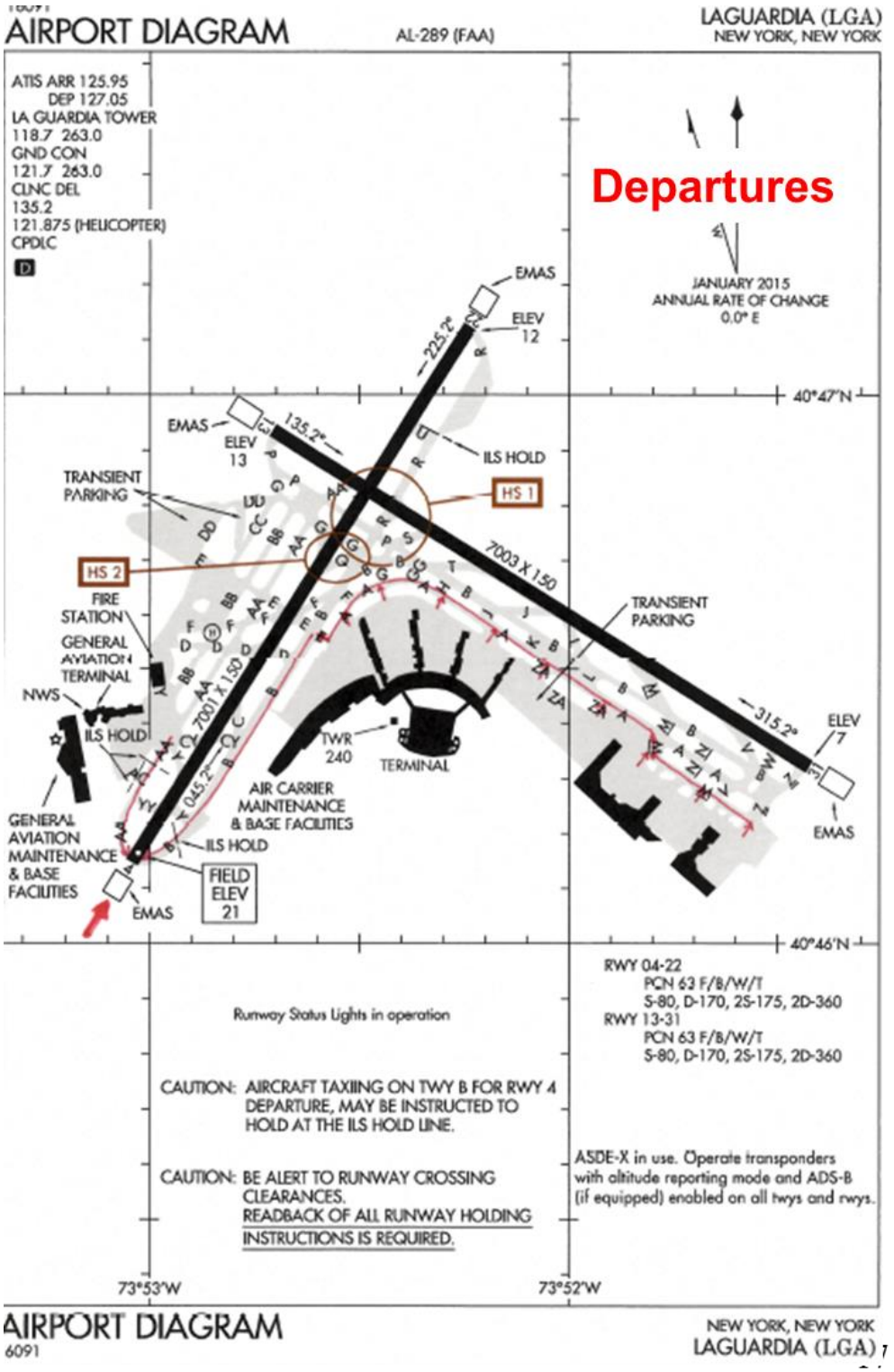


Figure 70. Illustration of LGA Surface Departure Routes

11.4.1 SME Review of LGA Surface Departure Routes

In order to better understand real-world influences that affect taxi-time for the surface routes modeled at LGA, we reviewed the illustration of the surface departure routes with our N90 SME's Ralph Tamburro and Bill Cotton. They offered the following comments on the taxi departure routes depicted in the illustration:

- This is a generally favorable configuration because many arrivals can exit 31 before crossing 04. One in, one out for crossing runway configuration.
- The alleyways between concourses in main terminal are used one way at a time because they are too narrow for passing. Taxiway Bravo has no parallel taxiway eliminating flexibility in sequencing takeoffs.
- Most aircraft make it off before the intersection – cross 04 as needed. 10-12 is typical queue length – LGA is busy all day. Bravo is always busy because of constraints on departures as well as one in one out.
- No gate hold at LGA. Narrow alleyways – one in one out. Gotta keep the alleys clear. So they end up with a large departure queue at the runway to keep gates open. All entrances to ramps are busy. All entryways to Alpha are busy. LGA is one of the more difficult airports for ground control. Planned improvements in terminal and ramp area (future) may alleviate some of the ground movement problems and enable the use of gate holds. Main terminal is being redesigned and should provide better manageability of traffic.

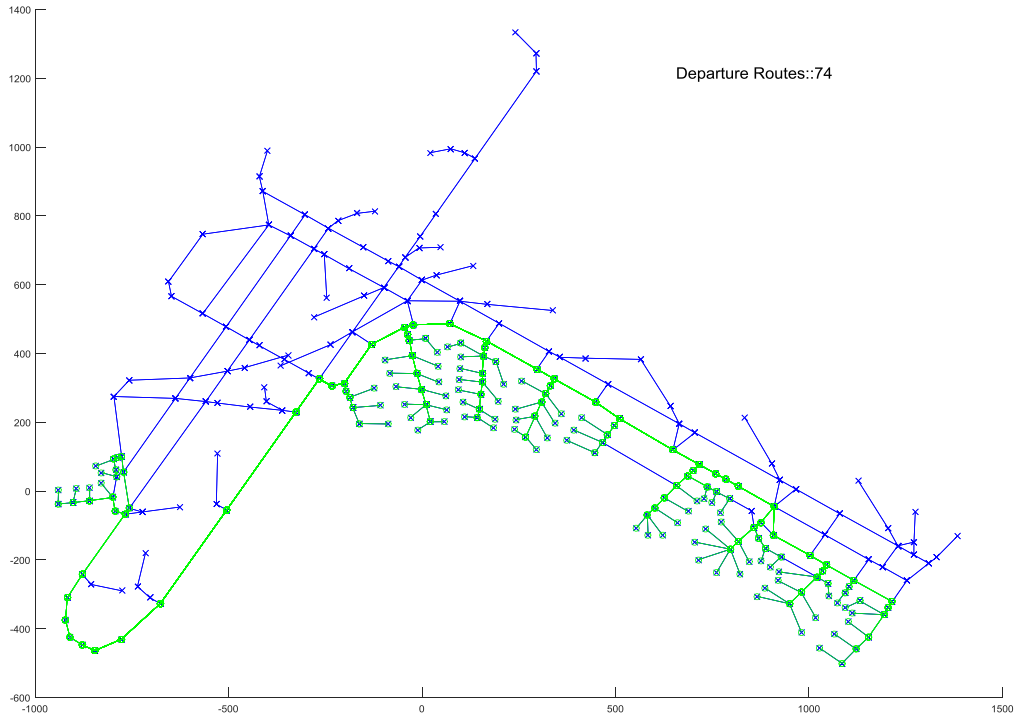


Figure 71. LGA Surface Departure Routes in SOSS

11.5 Runway Separation Requirements

In SOSS, separation matrices are used to specify the minimum required temporal separation between two successive operations on the same runway or a pair of interacting runways in the simulation. For JFK, we use the runway interactions and runway separation matrices defined in [SS12]. For EWR and LGA we use runway separation matrices that correspond to the runway interactions from the individual runway modeling data developed during the NASA ACES B3 development effort [ACES05]. Note that because SOSS does not delay arrivals, no separation matrix is specified for arrival-arrival pairs. Following are the runway interactions for each airport along with the runway separation matrices as implemented in SOSS in the separations.txt file.

11.5.1 JFK Runway Interactions and Runway Separation Matrices

JFK 31L, 31R|31L

DEP 31L – DEP31L

ARR31L – DEP31L

DEP31L - ARR31L

Airport JFK

#

Departure After Same Runway RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_RNAV_DEP 31L

dummy	S	L	H	B75
S	60	60	120	120
L	60	60	120	120
H	60	60	90	90
B75	60	60	90	90

Departure After Same Runway Non-RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_NONRNAV_DEP 31L

dummy	S	L	H	B75

S	60	60	120	120
L	60	60	120	120
H	60	60	90	90
B75	60	60	90	90

Departure After Coupled Runway Departure

Separation Matrix

column is leader and row is follower, WC enum is {H, L, B75, S}

31L and 31R are more than 4300 ft apart

Departure After Shared Runway Arrival

Operations Separation Matrix: separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SHARED_ARR 31L

dummy S		L	H	B75
S	50	60	70	60
L	50	60	70	60
H	50	60	70	60
B75	50	60	70	60

Departure Before Shared Runway Arrival

Operations Separation Matrix: separation[follower][leader]

x - departure, y - arrival, WC enum is {H, L, B75, S}

#

DEP_BEFORE_SHARED_ARR 31L

dummy S		L	H	B75
S	40	40	40	40
L	28	28	28	28
H	24	24	24	24
B75	28	28	28	28

11.5.2 EWR Runway Interactions and Runway Separation Matrices

EWR 4R|4L

ARR 04R – ARR 04R Table 23.1 Same Runway (Reduced Separation)

DEP 04L – DEP 04L Table 1.1 Same Runway

DEP 04L – ARR 04R Table 2.1 RWY Inter Parallel

ARR 04R – DEP 04L Table 2.1 RWY Inter Parallel

Airport EWR

#

Departure After Same Runway RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_RNAV_DEP 04L

dummy	S	L	H	B75
S	40	46	120	120
L	46	56	120	120
H	50	60	90	90
B75	46	56	120	120

Departure After Same Runway Non-RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_NONRNAV_DEP 04L

dummy	S	L	H	B75
S	40	46	120	120
L	46	56	120	120
H	50	60	90	90
B75	46	56	120	120

11.5.3 LGA Runway Interactions and Runway Separation Matrices

LGA 31|4

DEP 04 – DEP 04 Table 1.1 Same Runway

DEP 04 – ARR 31 Table 10.1 Crossing runways: Runway 1 4000-7000 ft., Runway 2 4000-7000 feet.

ARR 31 – DEP 04 Table 10.1 Crossing runways: Runway 1 4000-7000 ft., Runway 2 4000-7000 feet.

Airport LGA

#

Departure After Same Runway RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_RNAV_DEP 04

dummy	S	L	H	B75
S	40	46	120	120
L	46	56	120	120
H	50	60	90	90
B75	46	56	120	120

Departure After Same Runway Non-RNAV Departure

Separation Matrix, separation[follower][leader]

column is leader and row is follower, WC enum is {H, L, B75, S}

#

DEP_AFTER_SAME_NONRNAV_DEP 04

dummy	S	L	H	B75
S	40	46	120	120
L	46	56	120	120
H	50	60	90	90
B75	46	56	120	120