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, New York 13057 USA

TEL: (315) 445-0550

FAX: (315) 445-9401

Document Revision History

Version	Date	Author	Change Sections	Description
1	April 13, 2015	Aditya Saraf, Steven Stroiney, Valentino Felipe (Saab Sensis Corporation); Bruce Sawhill, Jim Herriot, Jim Phillips (NextGen Aerosciences Inc.)	All	Initial Release

Table of Contents

I. Executive Summary

THIS document is the final report for the NASA Aeronautics Research Institute (NARI)-sponsored project titled "Novel, Multi-disciplinary, Global Optimization Under Uncertainty". It outlines the research work done under a Phase I project between January, 2014 and January, 2015.

Under this research project Saab Sensis Corporation and its subcontractor, NextGen Aerosciences, developed a new, innovative, computationally-efficient framework for generating globally optimal solutions for complex system problems characterized by three key attributes: (i) presence of complex interactions between system components and network impacts, (ii) uncertain future system behavior, and (iii) multiple, competing, non-linear optimization objectives. We call this methodology PROCAST (Probabilistic Robust Optimization of Complex Aeronautics Systems Technology). One problem of significant national importance that displays the above three characteristics is the problem of managing air traffic in busy, highly interconnected regions of the National Airspace System (NAS)—namely, metroplexes. A metroplex consists of multiple busy airports in close vicinity of each other. Metroplex airports and terminal airspaces are key capacity bottlenecks within the national air transportation network. Applying the PROCAST methodology to this complex problem, we developed and assessed a prototype decision support tool (DST) for improving the management of metroplex air traffic on the airport surface as well as in the terminal airspace.

PROCAST is a multi-disciplinary "convergent" innovation because it combines for the first time the attributes of modern developments in the disciplines of Statistical Modeling (Data Science, rather) and Complex Adaptive Systems. From the first discipline (Statistical Modeling/Data Science), PROCAST uses Bayesian Belief Networks (BBNs)—a probabilistic graphical model (a type of a statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph. BBNs provide advantages over traditional probabilistic approaches, such as regression, by compactly encoding complex conditional distributions over a high-dimensional space and by providing a separation between solution method and model. From the second discipline (Complex Adaptive Systems), PROCAST uses NextGen AeroSciences' Continuous Re-planning Engine (NACRE)—a method of constrained optimization that leverages genetic algorithms. NACRE relies on continuous representation of 4-D (3 space dimensions plus time) paths that can be continuously and smoothly deformed between two given 4-D points.

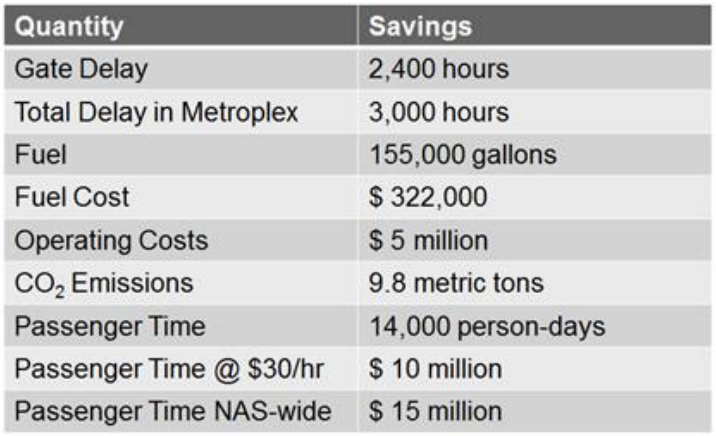
PROCAST solves complex system problems by using a simple, yet innovative, philosophy—(i) first, generate thousands of realistic potential futures, (ii) then, optimize over the entire domain (consisting of multiple interacting sub-domains) for each potential future, and (iii) finally, among the thousands of optimized solutions select the statistically best solution given the identified uncertainty models, by using clustering and statistical assessment. PROCAST uses BBNs for generating thousands of realistic forecasts of potential futures (e.g., future trajectory forecasts in the case of the metroplex traffic scheduling problem we address) based on the current measured state (e.g., positions and speeds of aircraft in the case of metroplex traffic scheduling) as well as by taking advantage of the knowledge "learned" by the BBNs based on historical recorded operational data. PROCAST uses NACRE for fast optimization over each future scenario (i.e., trajectory prediction in the case of metroplex traffic scheduling). In summary, using a simple philosophy PROCAST integrates BBNs and NACRE in an innovative manner to provide a decision support capability for facilitating optimal decisions under uncertainty.

In our Phase I research work, we demonstrted the viability of PROCAST by applying it to an air traffic management (ATM) test-problem. For this problem, PROCAST provides a multidisciplinary design approach for integrated arrival-departure-surface traffic scheduling, covering two flight domains: terminal airspace and airport surfaces. In Phase I, we conducted proof-of-concept simulation experiments that demonstrate the benefits of a PROCAST DST over current-day operations at a single-airport and terminal airspace (with plans for multi-airport traffic scheduling in Phase II work). We selected the busiest New York area airport—John F. Kennedy International Airport (JFK)—for this proof-of-concept demonstration. NASA's surface traffic simulation platform—the Surface Operations Simulator and Scheduler (SOSS)—was used to conduct the benefits analysis of PROCAST. To enable testing in an integrated airspace and surface domain, we enhanced SOSS to simulate terminal airspace traffic and to model pre-pushback operations.

The integration of different sub-components of PROCAST (i.e., individual software modules such as NACRE, BBNs and SOSS) produced by different sub-project teams into a single, unified system was a challenging task. There were three different development efforts progressing in parallel during the middle phase of the project—(i) developing a MATLAB-based simulation framework that included BBN-based prediction capability, (ii) enhancing

NASA’s SOSS simulation platform for extending its simulation capabilities to cover the terminal airspace, and (iii) developing the NACRE genetic algorithm (GA) for optimizing 4D trajectories. These three development processes were interdependent and were being developed in parallel by geographically distributed teams. We adopted an Agile integration process to ensure timely, correct, and seamless integration of these sub-components, as well as for managing changing modeling requirements. The Agile software integration process provided us with useful and timely feedback on correctness and integrability of different software components throughout the development process, and reduced time required for software development and integration.

We performed a simulation-based proof-of-concept demonstration of PROCAST’s benefits. Simulations showed that PROCAST can provide large delay and fuel savings benefits even when applied to a single-airport (JFK) problem. We used realistic and representative traffic scenarios, derived from real, airport and terminal airspace surveillance data, for these simulations. When converted to an annualized scale, these benefits amount to significant savings in delay, fuel, airline operating costs, emissions, and passenger time, as shown in Table 1. Far bigger benefits are expected when the PROCAST methodology is extended to the metroplex domain.



Quantity	Savings
Gate Delay	2,400 hours
Total Delay in Metroplex	3,000 hours
Fuel	155,000 gallons
Fuel Cost	$ 322,000
Operating Costs	$ 5 million
CO_2 Emissions	9.8 metric tons
Passenger Time	14,000 person-days
Passenger Time @ $30/hr	$ 10 million
Passenger Time NAS-wide	$ 15 million

Table 1. Summary of PROCAST’s estimated annualized benefits.

We developed a preliminary concept of operations for describing how a PROCAST methodology-based metroplex DST could be used collaborativelty by FAA and airlines. As a part of this concept of operations, we posit the existence of a new entity, referred to as the Metroplex Coordination Center (MCC), which will operate the PROCAST DST and ensure that the metroplex operates for the greater good of all participants. The MCC will periodically receive multiple inputs from different ATM systems, including (i) Expect Departure Clearance Times (EDCTs) for any active Ground Delay Programs (GDPs) that impact New York departures from the Traffic Flow Management System (TFMS), (ii) active miles-in-trail restrictions that may impact the New York TRACON arrival or departure traffic (again from the TFMS), (iii) arrival-fix crossing time estimates from the Traffic Management Advisor (TMA) or Time-Based Flow Management (TBFM) system at the local TRACON or center, (ii) acceptance rates and runway departure queue entry time estimates from the local airports, and (iii) pushback readiness time estimates, pre-pushback process state updates, and departure slot requests from airlines. The MCC will use the PROCAST system to generate airport-specific schedules for departures in the form of target take-off times (TTOTs) while rationing departures per departure-fix. The MCC may also use PROCAST to compute en route delays for arrival flights and communicate these delays to the TMA or TBFM system in the center. Further, using taxi conflict detection and resolution logic within NACRE, the MCC may also recommend taxi merge sequences to the ground controllers at the local airports.

Finally, although in our research we applied it to a trajectory optimization problem, the PROCAST methodology is directly applicable to a wider range of problems—e.g., traffic flow management decision making under uncertainty, real-time airport/airspace operations safety assessment, road traffic management, decision-making in healthcare, and others—and has strong potential for integrating into multiple existing NASA programs.

II. Project Background

AIRPORTS and terminal airspaces in busy metropolitan areas are key capacity bottlenecks within the national air transportation network. Two key aeronautics challenges relevant to traffic management in these constrained flight domains—as identified by the 2010 National R&D Plan [N10]—are: (i) developing the capability to perform four-dimensional trajectory (4DT)-based planning, and (ii) increasing airport approach, surface and departure capacity. The key to addressing both of these challenges simultaneously is to generate globally-optimal trajectory optimization and de-confliction strategies that span multiple flight domains of the National Airspace System (NAS), while efficiently addressing the strong and uncertain interactions between component subsystems. Given the pressing aeronautics challenges, NASA and other researchers are developing air traffic optimization concepts for alleviating capacity problems on the airport surface and in the terminal airspace. However, most of these research efforts approach the problems of uncertainty and interaction effects by applying piecewise control (i.e., separate planning functions for each flight domain) and the use of frequent re-planning on a local level, without a clear connection to their effects on overall system performance.

For instance, NASA’s Spot and Runway Departure Advisor (SARDA) research is developing planning and advisory tools primarily aimed at controlling gate pushback times, movement area entry times, and runway sequencing for departure flights in order to minimize taxi times and maximize departure throughput. Similarly, NASA terminal and metroplex operations research efforts have applied a piecewise approach to the arrival traffic management problem—Traffic Management Advisor (TMA) and Efficient Descent Advisor (EDA) perform en route descent trajectory management [NE90, GV01], Final Approach Spacing Tool (FAST) performs planning within the TRACON [IR01], Interval Management (IM) addresses spacing operations in en route and terminal areas [NB09], and the metroplex scheduling concept [SC11] applies to multi-airport arrival management from the arrival-fix to the landing runway. NASA and industry benefits assessment efforts [CH12, CM07, SC11] have shown that, taken individually, the ATM concepts described above provide valuable benefits. However, efforts to directly interface the surface and terminal traffic management concepts with each other in their current forms may not (and likely will not) yield globally-optimal and computationally tractable solutions because of challenges posed by complex subsystem interactions, uncertainty impacts and network effects.

Furthermore, most current DST approaches rely on frequent re-planning to address uncertainty. The expectation is that the use of real-time information for updating the plans frequently would alleviate the impacts of uncertainty. However, owing to the large uncertainties associated with airport surface traffic, this approach may not work very well in the real-world. For example, suppose that SARDA decides to hold flight A at its gate to let flight B depart ahead of it given the predicted traffic situation. Say, five minutes later (when flight B was predicted to pushback), SARDA realizes that flight B has incurred a pre-pushback delay, and now flight A can depart ahead of B. At this point in time, re-planning cannot revoke the 5-minute delay already taken at the gate by flight A. Instead, if the planning function had used information on flight B’s likelihood of getting delayed prior to its pushback, it may have made a better decision. Moreover, with frequent re-planning comes the additional disadvantage of potential instability in successive solutions. The alternative approach of incorporating uncertainty directly in the optimization problem may yield robust solutions. However, for ATM problems that span multiple flight domains, incorporating uncertainty into models involves probabilistic evaluations over a large number of dependent random variables. Traditional approaches do not provide computationally tractable methods for this problem. The key is to develop a mechanism for exploiting the structure in the random variable distributions. By identifying the cause and effect relationships and using them to develop a compact representation of probabilistic dependencies, it is possible to represent a dauntingly large number of distributions compactly, and in a way that allows these distributions to be constructed efficiently from recorded or simulated data. This model can be utilized effectively to calculate conditional probabilities for key optimization parameters.

The Saab-led research project titled “Novel, Multi-disciplinary Global Optimization under Uncertainty,” developed a novel multidisciplinary design approach, namely PROCAST, for addressing the problems discussed above and applied it to the problem of integrated arrival-departure-surface traffic scheduling. The PROCAST multidisciplinary optimization accomplishes two objectives: (i) unification of the dynamics of traffic in multiple flight-domains—ultra-dense terminal airspace and the ground traffic associated with that terminal airspace; and (ii) probabilistic, fast-time re-planning of trajectories. With regard to the former objective, unification increases the number of degrees-of-freedom (DOFs) in the optimization problem, thereby making the solution-search more challenging; but the scope of the solution space is also expanded, allowing for more optimal solutions in the combined spaces (air and ground) than is possible if they are considered separately. With regard to the second objective, air and ground traffic will always have a degree of unpredictability due to system complexity, winds, weather, aircraft performance variance, human decisions and activities, and uncertainty in flight event time predictions. This uncertainty means that for the solution to be robust, the schedule of trajectories must be adaptable to changing situations, while maintaining safe spacing between flights and still remaining close to optimal.

To achieve these objectives, PROCAST combines the attributes of modern developments in the disciplines of Statistical Modeling (or Data Science) and Complex Adaptive Systems. From the first discipline (Statistical Modeling/Data Science), PROCAST uses BBNs—a probabilistic graphical model (a type of a statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph. BBNs provide advantages over traditional probabilistic approaches, such as regression, by compactly encoding complex conditional distributions over a high-dimensional space and by providing a separation between solution method and model. From the second discipline (Complex Adaptive Systems), PROCAST uses NextGen AeroSciences' Continuous Re-planning Engine (NACRE)—a method of constrained optimization that leverages genetic algorithms. NACRE relies on continuous representation of 4-D (3 space dimensions plus time) paths that can be continuously and smoothly deformed between two given 4-D points.

PROCAST solves complex system problems by using a simple, yet innovative, philosophy—(i) first, generate thousands of realistic potential futures, (ii) then, optimize over the entire domain (consisting of multiple interacting sub-domains) for each potential future, and (iii) finally, among the thousands of optimized solutions, select the statistically best solution given the identified uncertainty models, by using clustering and statistical assessment. PROCAST uses BBNs for generating thousands of realistic forecasts of potential futures (e.g., future trajectory forecasts, in the case of the metroplex traffic scheduling problem we address) based on the current measured state (e.g., positions and speeds of aircraft, in the case of metroplex traffic scheduling) as well as by taking advantage of the knowledge "learned" by the BBNs based on historical recorded operational data. PROCAST uses NACRE for fast optimization over each future scenario (i.e., trajectory forecast, in the case of metroplex traffic scheduling). In summary, using a simple philosophy, PROCAST integrates BBNs and NACRE in an innovative manner to provide a decision support capability for facilitating optimal decisions under uncertainty.

The rest of this report is structured as follows. Section III describes the PROCAST solution architecture—namely, the technical approach that we developed for integrating BBN-based probabilistic forecasts directly into NACRE-based 4D trajectory optimization. Sections IV and V provide an overview of BBNs and NACRE, respectively. Section VI introduces two supporting capabilities for simulation and integration, which are described in more detail in Appendices A and B, respectively. Section VII describes our experiment design for PROCAST proof-of-concept demonstration. Section VIII presents results obtained from the proof-of-concept simulations. Section IX discusses a preliminary concept of operations for a PROCAST-based decision support tool. Finally, Section X outlines the conclusions from the Phase I research work and discusses proposed future work.

III. PROCAST Solution Architecture

THE aim of PROCAST is to provide optimized (fuel or time) and de-conflicted 4D surface-terminal flight trajectories using a planning function that is robust to uncertainty in forecasts, as well as to the imprecision in execution of trajectory controls. To achieve optimization with robustness, PROCAST works on a simple philosophy—each time a plan is generated or updated, PROCAST first uses BBNs (trained on historical data and conditioned on the current measured state of traffic) to generate multiple (thousands) potential future trajectory forecasts/scenarios. Then, using NACRE, PROCAST performs trajectory optimization on each future scenario. Once optimization over a large number of scenarios is complete, probabilistic analysis is used to compute the "statistically-best" set of optimal control actions (i.e., flight release times), and this set of actions is chosen for implementation on the traffic. A short time later, this process is repeated to update the plan using the latest available information.

In keeping with this philosophy, as shown in Figure 1, PROCAST first generates thousands of potential future traffic scenarios. Each potential future traffic scenario consists of a set of unimpeded flight trajectories for each flight that is currently active within the terminal area or is expected to enter it within a certain look-ahead time horizon (by unimpeded we mean trajectories flown or traversed without any de-confliction with respect to other traffic). For generating future trajectories, PROCAST assumes that the physical route of the flight will remain fixed—only variations in the time of arrival/crossing at key nodes on the fixed routes are introduced for generating multiple futures. The time-variations are generated based on intelligent sampling from the space of all possible futures using probabilistic models derived from historical data-mining (i.e., BBNs).

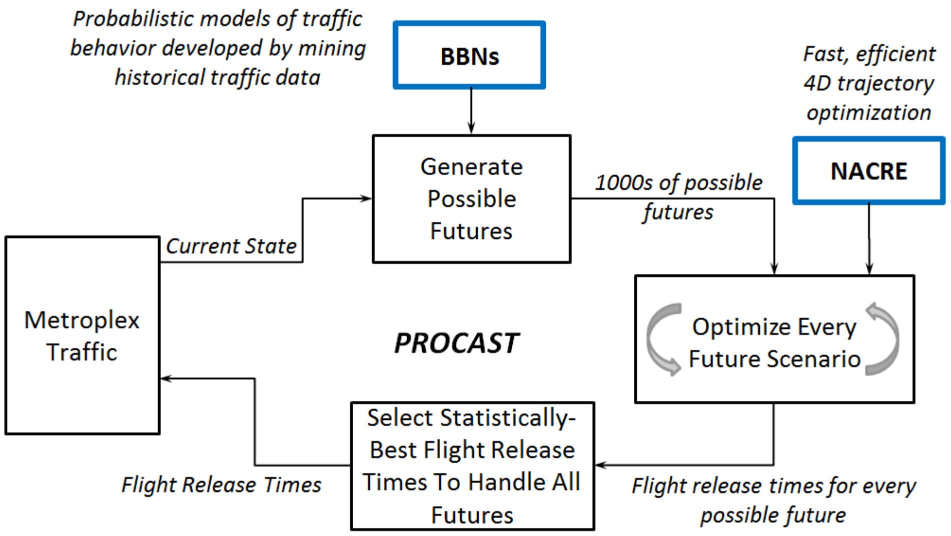

Probabilistic models of traffic behavior developed by mining historical traffic data
BBNs
Fast, efficient 4D trajectory optimization
NACRE
Generate Possible Futures
1000s of possible futures
Current State
Metroplex Traffic
PROCAST
Optimize Every Future Scenario
Select Statistically-Best Flight Release Times To Handle All Futures
Flight Release Times
Flight release times for every possible future

Figure 1. PROCAST performs thousands of cycles of 4D trajectory optimization, each starting with a randomly sampled potential future. Probability distributions, from which PROCAST samples intelligently to create potential futures, are generated by BBNs taking advantage of "learning" from historical data and measurement of the current state of traffic.

Figure 2 shows how randomness is added to the "time" dimension of the 4D trajectory representation at key nodes or waypoints using pre-computed probability distributions. The probability distributions shown in the figure describe the probability of a flight arriving at certain fixed 3D nodes at time 't'. The probability distributions are computed using a BBN that is trained using historical trajectory data and are conditioned on several influencing factors including the current measured state of the airport-terminal system (i.e., locations and speeds of all aircraft). The key nodes/waypoints at which we add time-randomness are gates, spots, runway departure nodes, airspace interaction fixes, and departure/arrival fixes, some of which are shown in Figure 2.

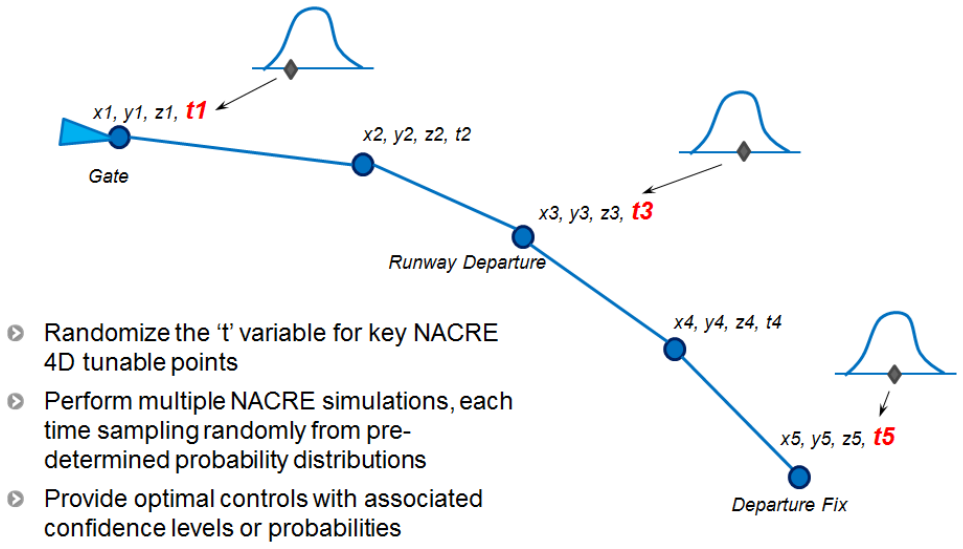

x1, y1, z1, t1
Gate
x2, y2, z2, t2
x3, y3, z3, t3
Runway Departure
x4, y4, z4, t4
x5, y5, z5, t5
Departure Fix
Randomize the 't' variable for key NACRE 4D tunable points
Perform multiple NACRE simulations, each time sampling randomly from pre-determined probability distributions
Provide optimal controls with associated confidence levels or probabilities

Figure 2. PROCAST randomizes the time-dimension of the 4D trajectory representation used by NACRE.

After generating thousands of potential futures, PROCAST performs 4D trajectory optimization/de-confliction using a 4D trajectory optimizer (NACRE) for each of the potential futures (numbering in the 1000s). In each cycle, the optimization acts on one potential future traffic scenario (covering the next 45 minutes) and produces a set of optimal controls that de-conflict the traffic with minimum costs (fuel/time/hybrid), given that particular scenario. Optimization controls are flight release times at certain key nodes—gate, spot, runway, departure fix for departures, and arrival-fix, runway, and spot for arrival flights. Each optimization cycle produces a set of optimal control actions, to which we allocate a probability equal to the probability of occurrence of the associated potential future. After completing thousands of NACRE optimization cycles, we perform statistical assessment of the occurrence-rate of each prescribed control action computed by NACRE during the optimization cycles, and select the “statistically-best” control actions for implementation.

In our Phase I work, we use a NASA simulation platform called Surface Operations Simulator and Scheduler (SOSS) as the model of a real airport and terminal airspace. Hence, the “statistically-best” control actions are sent to the simulator at the end of each optimization cycle for implementation. This process is repeated periodically to update the plan using the latest available information.

In summary, the proposed randomized, multi-cycle optimization approach provides a rapid and automated ‘what-if’ analysis capability to facilitate optimized traffic management under uncertainty. BBNs provide a method to characterize, in fast-time, the probabilities of thousands of possible futures conditioned on the current state of the airport surface and terminal airspace traffic. NACRE provides the capability to perform multiple iterations of 4D trajectory optimization rapidly, each time with the assumption of a different possible future.

The next two sections provide an overview of BBNs and NACRE, respectively.

IV. Bayesian Belief Networks and Their Role in PROCAST

ONE of the main goals of the research presented here is to make surface-terminal operations planning robust to uncertainties such as unpredictable errors in the forecasts of aircraft gate pushback times, taxi times, and terminal airspace traversal times. To achieve this goal, we make use of BBNs for incorporating uncertainty models into trajectory optimization in a computationally efficient manner. A large number of random variables are required to sufficiently model uncertain network effects in the surface-terminal operations. Moreover, 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 prohibitively large number of conditional probability distributions (CPDs). BBNs provide a mechanism for exploiting the structure in complex distributions to describe them compactly so that they can be constructed and utilized effectively [KF10].

BBNs graphically model causes and their effects in a probabilistic manner [KF10]. A BBN consists of a set of nodes, which represent random variables, and a set of directed links, which model the conditional dependencies between the random variables. Each node is associated with a probability function which takes as input a set of values for the node’s parent nodes and gives the consequent CPD of the random variable associated with the node. A BBN is a directed acyclic graph. Cyclic graphs are called Markov Networks (MNs). The conditional dependencies between random variables in the graphical model are quantified by CPDs, which can be computed from historical data or simulation data. In our work we use a BBN for predicting the transit of a flight through the surface-terminal network.

A. Bayesian Belief Networks Explained

This section provides a short background on BBNs for those who are not familiar with these models. To explain BBNs, let us consider a very simple example where we are interested in finding the probability that a flight departs from an airport with less than 15 minutes of delay. We might model this with a binary-valued random variable (RV), *Delay15* (True/False). Suppose we also have a two-valued RV for the *weather condition* (Visual or Instrument Meteorological Conditions—VMC or IMC); a four-valued RV for the *runway configuration* (RC1, RC2, RC3, RC4); a two-valued RV for two possible *taxi route* options (T1 and T2); a two-valued RV for two possible *departure runways* (R1 and R2); and a two-valued RV for modeling *runway congestion* (True/False). Overall, our probability space has 2 X 2 X 4 X 2 X 2 X 2 = 128 values, corresponding to the possible assignments to these six RVs. If we have a joint distribution over all these RVs, we can answer questions such as how likely is the flight to have less than 15 minutes of delay given IMC weather, RC3 configuration, T2 route option, and so on. Specifying a joint distribution over 128 possible values already seems daunting, even for such a small problem.

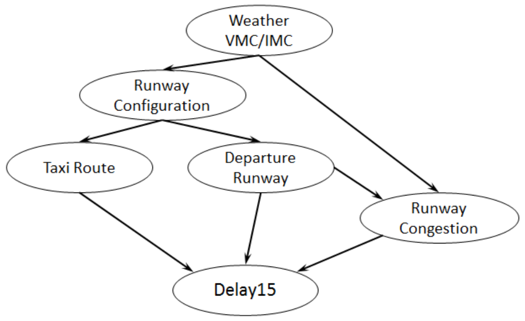


Figure 3. BBN example illustrates the power of compact CPD representation (Note: This is an illustrative example and not the BBN used in PROCAST).

To overcome the curse of dimensionality, BBNs use a graph-based representation of cause-and-effect relationships among the random variables, as shown in Figure 3, for compactly encoding a complex distribution over a high-dimensional space. The nodes correspond to the RVs, and the edges correspond to direct probabilistic interactions between them. With this graphical representation, we see that instead of encoding the probability assignment for every possible assignment to all the RVs, we can break-up the distribution into smaller factors, each over a smaller space of probabilities. We can then define the overall joint distribution as a product of these factors. For example, the probability of the event "*Delay15* = True (i.e., Delay < 15 minutes) in IMC *weather*, RC3 *configuration*, T2 *taxi route*, R1 *departure runway*, no *runway congestion*" can be computed by multiplying six numbers: P(*Weather* = IMC), P(*RC* = RC3 | *Weather* = IMC), P(*Taxi Route* = T2 | *RC* = RC3), P(*Departure Runway* = R1 | *RC* = RC3), P(*Congestion* = False | *Departure Runway* = R1, *Weather* = IMC), and P(*Delay15* = True | *Taxi Route* = T2, *Departure Runway* = R1, *Congestion* = False). This parameterization is significantly more compact, requiring only 1 + 2 + 4 + 4 + 4 + 8 = 23 non-redundant parameters—for characterizing the following smaller factors: P[*Weather*] (1 parameter), P[*RC* | *Weather*] (2 parameters), P[*Taxi Route* | *RC*] (4 parameters), P[*Departure Runway* | *RC*] (4 parameters), P[*Congestion* | *Weather*, *Departure Runway*] (4 parameters), and P[*Delay15* | *Taxi Route*, *Departure Runway*, *Congestion*] (8 parameters). In summary, BBNs eliminate unnecessary DOFs to compactly represent the joint distribution and provide fast computations of the required CPDs.

For use in PROCAST, we harness the compact-representation power of BBNs to efficiently develop time-of-arrival probability distributions for key waypoints/ nodes in the surface-terminal network. We discuss our technical approach for generating BBNs in the next section.

B. Predicting Surface-Terminal Trajectories Using BBNs

The specialized function of BBNs in the PROCAST architecture is to create thousands of potential future trajectory scenarios, utilizing the measurement (full or partial) of the current state of the metroplex traffic and the historical "learning". We developed BBNs to cover trajectory predictions over two different phases of departure flight—pre-pushback and airport surface transit. Currently, we do not utilize a BBN to cover predictions for transit times in the terminal-airspace and on the airport surface for arrivals. We use deterministic transit time predictions for these parts of the flight's trajectory. However, our approach could be enhanced in Phase II to include probabilistic modeling for these parts of the trajectory, as well.

We have two types of BBNs. Type 1 BBN predicts gate pushback time of a departure flight based on either (i) the current state of its pre-pushback process, if it is waiting at the gate; or (ii) based on the state of the previous connecting flight, if the prior flight has not landed or reached its gate yet. Type 2 BBN predicts the transit of a departure flight along the airport surface. The type 2 BBN, in fact, is a group of multiple inter-dependent BBNs. We created one BBN per spot-to-departure-runway pair. Each type-2 BBN predicts the times of arrival at/release from important transit points in the flight's route, e.g., gate, spot, key taxiway merge-node(s) (these are important transition points where aircraft merge and then join a common departure queue), and runway entry. Some nodes in the BBN correspond to random variables signifying the times at which the aircraft cross these important transit points. Other nodes represent influencing factors such as the actual/predicted length of the runway departure queue at the actual/predicted time of the aircraft's pushback.

Pre-pushback Prediction BBN (Type 1): The pre-pushback prediction (type 1) BBN predicts a probability distribution of gate pushback readiness time, based on the current state of the pre-pushback process. We use simulated pre-pushback process model data for training the BBN. Figure 4 shows the pre-pushback simulation model used in this work. The turnaround process is broken down into three sub-processes—de-boarding, fuelling, and boarding. Although the actual turnaround process consists of other sub-processes such as cleaning and catering, these processes usually occur in parallel with one of the three modeled sub-processes. The reason for selecting only three modeled processes is that these are the only processes in the critical path of the turnaround process [BT12]. The modeling works as follows—after the entry of the prior connecting arrival flight into the gate, there is a random time-interval (ΔT1) that accounts for pre de-boarding activities. After ΔT1 the de-boarding process starts. De-boarding continues for a certain time interval (ΔT2). After de-boarding finishes there follows an interval time ΔT3 to account for pre fuelling activities, after which fuelling starts. Fuelling continues for a time-interval ΔT4. After fuelling finishes there follows an interval time ΔT5 to account for pre boarding activities, after which boarding starts. Boarding continues for a time-interval ΔT6. The flight is deemed ready for pushback at the end of the boarding process. The inter-subprocess time-intervals (ΔT1, ΔT3, and ΔT5) are modeled as normally distributed random variables with parameters derived from real-world observations of turnaround operations in prior work [BT12]. The time-durations of the three sub-processes (ΔT2, ΔT4, and ΔT6) are modeled using Weibull distribution as per analysis in [BT12].

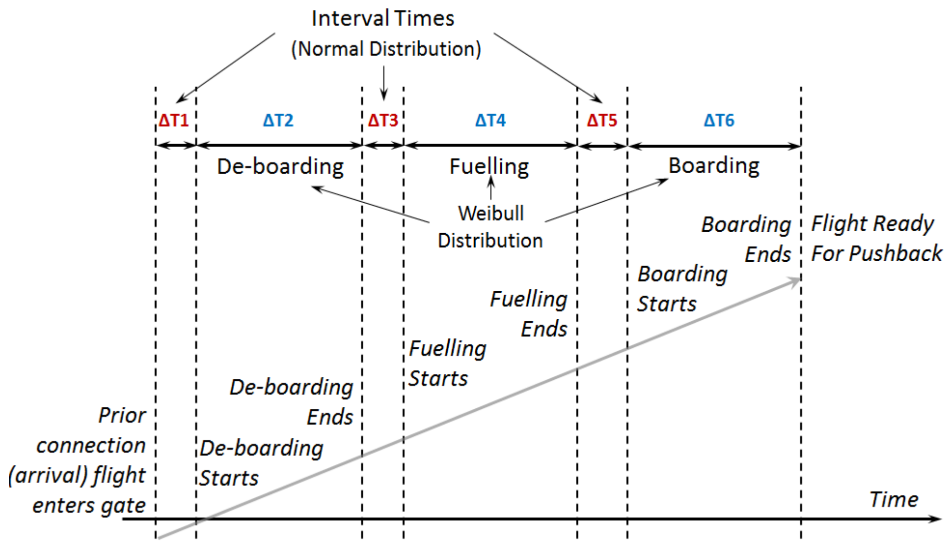


Figure 4. Pre-pushback process simulation model.

Training data for the pre-pushback prediction BBN is generated by conducting Monte Carlo simulations of the turnaround process model. It is assumed that periodic updates regarding where each departure flight roughly is in its turnaround process will be available to the BBN in the form of the current completion level of each of the sub-processes. For example, the current state of pre-pushback process for a flight could be *% Boarding Complete* = 100, *% Fuelling Complete* = 50, *% Boarding Complete* = 0. Using this information about the current state of the pre-pushback process, the BBN predicts the estimated gate pushback readiness time. The CPDs involved in this BBN are of the following form—Prob [gate pushback time = *t*+*k* | At t we have De-boarding % *p1* complete, fuelling % *p2* complete and Boarding % *p3* complete], for all possible values of *k*, and valid combinations of *p1*, *p2*, and *p3*. These CPDs are derived from the data created by the Monte Carlo simulations. Figure 5 shows the simple BBN used for predicting a flight’s pushback readiness time using a measurement of its pre-pushback process state. We used a simple prediction model, since we did not have detailed airline pre-pushback data. In Phase II project work, we aim to improve on the pre-pushback modeling by leveraging detailed airline data as well as airline subject matter experts to define a more complex (and hopefully more accurate) pushback readiness time prediction BBN.

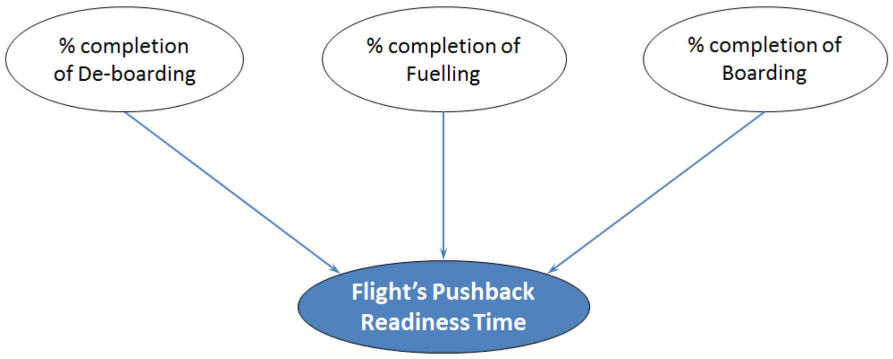


Figure 5. Simple BBN for modeling impact of pre-pushback process state on pushback readiness time.

Surface Transit Prediction BBN (Type 2): Figure 6 shows a simplified version of the airport surface transit prediction (type 2) BBN that we developed for predicting key event times for departure traffic going from one terminal-gate area to one departure runway.

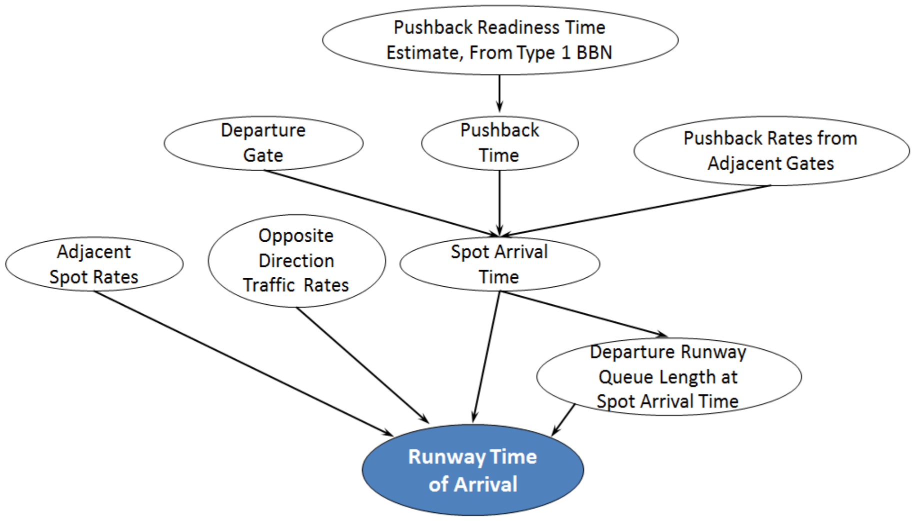

Pushback Readiness Time Estimate, From Type 1 BBN
Departure Gate
Pushback Time
Pushback Rates from Adjacent Gates
Adjacent Spot Rates
Opposite Direction Traffic Rates
Spot Arrival Time
Departure Runway Queue Length at Spot Arrival Time
Runway Time of Arrival

Figure 6. Simple BBN for departure traffic prediction from one terminal-gate area to one departure runway.

The random variable and interconnections in our model are different from those shown in this simplified model (real model not shown because it is too complicated to include in a single figure). We derive the CPDs with each link in the graph from simulation data. For example, the interrelationship between “Pushback Time” and “Spot Arrival Time” is defined by the following set of CPDs: Prob[Spot Arrival Time = $t+k$ | Pushback Time = t, Gate = A, Pushback Rate From Adjacent Gates = n], , for all values of ‘k’, ‘A’, and ‘n’ observed in the simulation data.

Predicting Future Trajectories Using Type 1 and 2 BBNs: A combination of type1 and type 2 BBNs is used to generate multiple realistic potential future trajectories—starting with the generation of a probability distribution for gate pushback readiness time using the type 1 BBN. Following this, we randomly sample from the gate pushback readiness time probability distribution. This sample forms the input to the pushback time node in the Type 2 BBN. Values for different influencing factors (nodes) within the type 2 BBN are then estimated based on the current measurement of the airport-surface traffic state. Using these values, the BBN then estimates probability distributions for different variables of interest from a trajectory time-point perspective, i.e., Spot Arrival Time, Runway Arrival Time, etc. As a flight progresses through its surface transit, some of the nodes in the BBN are assigned fixed values—for example, “Pushback time” will be assigned a fixed value (i.e., the actual time of pushback) after the flight pushes back from its gate—thereby reducing the uncertainty in the downstream predictions.

C. Process for Generating BBNs for Surface-Terminal Operations

Our approach for generating BBNs consists of multiple sub-steps. First, subject matter experts on our team utilize their extensive surface and terminal operations know-how to select key random variables for inclusion in the uncertainty models and develop a graphical model of causal or correlational interactions between these variables. See Figure 5 and Figure 6 for type 1 and type 2 BBN graphical models; the actual type 2 model is similar in principle to Figure 6 but differs in the choice of nodal variables and their interconnections. The MATLAB Bayes Net toolbox [M01] is used to model the structure of the BBNs.

After the initial graphical model is ready, we next develop CPDs to characterize the causal or correlational relationships associated with each edge in the graph. For generating the CPDs (i.e., for "training" the model), we utilize an extensive simulation data archive. We did not have access to real airline pre-pushback process data, for generating parameters of the type 1 BBN; we utilize data generated by simulation using the model shown in Figure 4. For generating parameters of the type 2 BBN, we use simulated SOSS airport surface traffic data from thousands of airport simulation runs—each run was conducted by adding random perturbations to key flight schedule scenario parameters (e.g., scheduled gate pushback times), node/waypoint arrival times (e.g., simulated actual spot arrival time) or inter-waypoint/node transit times (e.g., simulated ramp taxi times) as modeled by random variations in NASA's simulation platform. We decided to use simulation data, instead of historical, recorded track data for training the type 2 BBN, because the eventual tests of the PROCAST scheduling concept will be conducted via simulations; and recorded data from actual operations usually do not compare well with simulated data due to modeling limitations. So, we do not want our BBNs to be trained on data that would differ from the simulated uncertainties.

After we generate the CPDs, the BBNs are ready to use for predicting probabilities such as the probability that flight *X* arrives at runway *Y* at time *t*. Although the prediction-computations may involve multiple dependent, random variables, they are very fast because, as we explained above, probability calculations only involve multiplying the component factors, not explicit enumeration over all possible combinations of random variable values. Next, we validate the BBN's predictions against simulation data. For validation, we select multiple days of operations that fall out of the range of the model training data. We select key prediction variables such as time of arrival at a runway as test-cases for validation. We perform probability prediction for these test-case variables using the BBN. We check the mean values of the probability predictions against the actual values observed in simulation. We also verify that the standard deviation of the probability distributions decrease towards zero as the prediction look-ahead time goes down, i.e., as the aircraft moves closer and closer to the prediction point. If discrepancies are discovered, we add/remove certain random variables (i.e., nodes) from the BBN, modify the cause-and-effect interactions (i.e., links), as well as re-calibrate the factor probabilities (i.e., CPDs) in the BBN to increase accuracy of predictions. Multiple design-modification and verification iterations are required to obtain an accurate predictive model.

V. NACRE—Genetic Algorithm for 4D Trajectory Optimization

NACRE was originally developed for dynamic replanning of 4D trajectories in the en route airspace, incorporating frequent status updates (on the order of a minute) of aircraft positions and velocities as well as frequent updates of wind, weather and other external factors to generate continuously-updated, optimal, de-conflicted trajectory plans. The original goal of NACRE was to maintain the enroute airspace in a state of "planned deconfliction," a condition in which all of the trajectories in the airspace were in a predicted conflict-free state out to the limits of deterministic knowledge, generally about 20-30 minutes. This was accomplished by a combination of efficient heuristics and frequent replanning.

NACRE uses a "co-operative agent" heuristic in which multiple computational agents are assigned the task of finding optimal conflict-free trajectories, one agent per trajectory, architecture well suited to parallelization. Since the search space of possible trajectories is enormous and far beyond the power of exhaustive search for all but the simplest problems, computational heuristics are required. A combination of ant colony optimization algorithms and differential evolution (a flavor of genetic algorithms suited to continuous deformation of trajectories) has been implemented. Cooperative agents "share notes" with each other in the process of solution finding, thereby more efficiently searching the solution space. In addition, they "keep notes" on solutions that have already been found, and this provides a computational speedup when searching for new solutions to closely related problems, as it provides a beneficially biased starting point for a new optimization. A cooperative agent approach is also well suited to a dynamic problem in which the problem description is continuously changing with new updates from the system being modeled, but where many aspects of the previous solution are still close to optimal and therefore relevant.

For this project, the interaction with the BBNs called for a reformulation of NACRE specific to ground traffic, as the BBNs generate a large number of scenario variants (on the order of thousands), each one requiring a solution from NACRE in a very short time. Generating solutions at this speed requires a simplification of NACRE that is less general than the airborne version but is capable of faster computation. The original version of NACRE is optimized for speed and written in C#, but the ground-specific version is written in MATLAB for the purpose of expeditious integration with the NASA simulation, SOSS. SOSS allows for plugging in external schedulers, such as PROCAST.

The airport surface is represented by a node-link model that captures a significant amount of fine detail (JFK is mapped onto 809 nodes, for instance). The optimization works with a set of pre-defined spatial trajectories that are characterized by an earliest time of departure from the gate or arrival at the terminal fix. The Phase I effort does not consider variable surface pathways, and the terminal air 3D trajectory is similarly constrained; so vectoring is not considered. The optimization takes place entirely in the time domain, using the implementation of delays at the gate (for departure flights) or beyond the arrival fix (for arriving flights).

Previous research has attacked the airport surface traffic problem with a variety of techniques [Be94,Ha01,Ki13]. Mixed-integer linear programming (MILP) produces excellent results for smaller problems, but becomes prohibitively time consuming for larger ones [AM10, GM09]. Faster results have been obtained with dynamic programming approaches, but these are not easily parallelizable, which is likely to be a future concern [Ba07,Ra09]. In addition, any of these other approaches are not well suited to continuously updated information, as each new problem configuration requires an optimization restart. Other heuristics have produced more adaptive results, such as genetic algorithms and ant colony optimization, but have not been combined with simulation-based updates to drive an adaptive optimization loop; nor have they considered coupling with the traffic in the terminal airspace [GD03, HD08, LW11]. This work builds on this previous work and adapts it to mixed airborne and surface operations.

A. NACRE Approach

NACRE divides the problem of airport surface traffic management into two parts—the first part is concerned with the optimization of runway system utilization, the second part is concerned with generating surface trajectories that are completely de-conflicted in time and space.

Since the most important aspect of an airport is its runways, we started with optimization of runway usage. Given a set of aircraft with desired departure and arrival times, the task became to sequence them optimally, respecting wake vortex considerations (weight classes of aircraft), departure fixes, and mixed operations on the same runway. Once an optimal sequencing with respect to a cost function (throughput or sum of squares of delays) was completed, the ground movements could be coordinated so as to match this runway sequence. For this phase of the research, surface trajectories traversed a pre-defined set of nodes. In order to assign a relative time to each node for the purpose of evaluating potential conflict, the SOSS simulation was used in an “unimpeded” mode, where the aircraft/trajectory combination in question was simulated without any other traffic.

Genetic Algorithm: The runway sequence was optimized using a genetic algorithm. Variants of this have been used before, but with an emphasis on alternate paths on the tarmac. The “genome” was a sequence of displacements equal in length to the number of aircraft being sequenced, with each displacement indicating how many places an aircraft was moved from its initial scheduled position. Casting mutations in the form of displacements was done with a view towards future metroplex terminal airspace planning, as aircraft approaching a fix or merge point and a potential conflict have a limitied amount of “wiggle room” to employ without vectoring.

These sequencing genomes were constructed in such a way that they could be subject to both mutation and crossover operations. As is standard with genetic programming, the search was begun with an initial population of displacements in different positions in the aircraft sequence, and the crossover operator would combine different sets of displacements from different genomes. Since the interaction with PROCAST and SOSS required frequent re-optimization as operations information was updated, we used a truncated search space where we did not consider all possible aircraft sequences, but rather limited the number of positions an aircraft could be displaced per re-optimization. The effect of truncating the search space is shown in Figure 7.

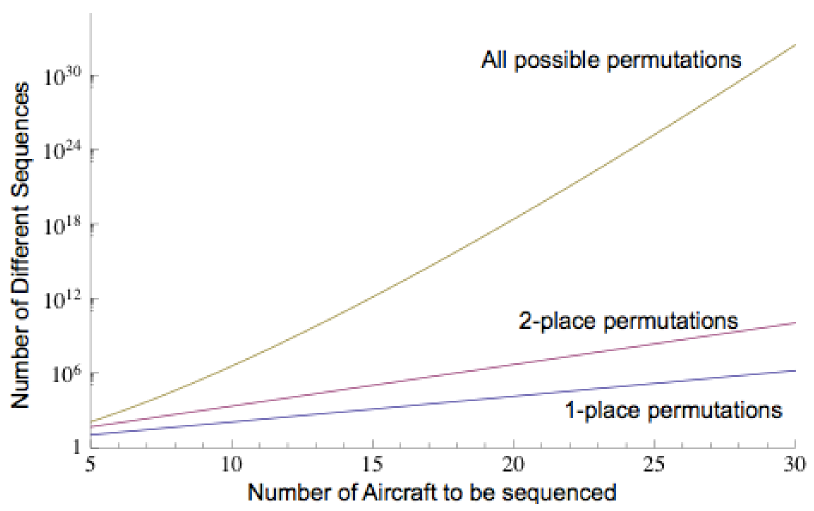


Figure 7. Search space, total (yellow) and under different truncations—2-place permutations (red) and 1-place permutations (blue).

For each round of optimization, the evolution proceeded until the population mix stabilized, indicating a saturation of the search process and the discovery of robust solutions (where robustness is understood as many different evolutionary pathways leading to the same results). This genetic process does not necessarily produce the provably lowest cost/highest fitness solution, but it produces high quality solutions quickly in cases where the search problem “factorizes.” Factorization in this context means that partial solutions (such as the second half of a sequence) generally interact weakly with other parts of the solutions (such as the first half of a sequence), so a good solution can be assembled using good sub-solutions.

Surface Deconfliction: For this project, we sequenced two runways at JFK, 31L and 31R, with one supporting mixed operations and the other arrivals-only. After we obtained good sequences for the runways, we computed backwards in time to obtain space-time sequences for the departing aircraft approaching the runways so as to meet the scheduled runway times. When a conflict was detected (two aircraft at the same place and the same time modulo a tolerance time) the conflict was resolved by a 4DT “rigid shift”—the entire surface trajectory was time-shifted by an amount enough to resolve the conflict. New conflicts generated by this process were solved recursively until no conflicts remained between departing aircraft or a departing aircraft and an arriving aircraft. Conflicts between two arriving aircraft between the runway and their gates were not considered as a part of Phase I, and all time shifts were implemented at the gate for departing aircraft or outside of the arrival fix for arriving aircraft. The effect of a 4DT rigid shift was to change the gate release time for a departing aircraft, and this aircraft would (in principle) not stop between pushback and arrival in the departure queue.

Because of the effectiveness of the genetic algorithm and the simplified surface dynamics, NACRE was able to compute runway sequencings and gate release times very quickly, on the order of a second for several dozen aircraft in the time horizons being considered. The results of each optimization computation were sent to PROCAST, which cycled the SOSS simulation forward a simulated amount of time (usually five minutes) and then returned a new set of desired arrival/departure times (with input from the BBNs) to be sequenced. This was repeated until an entire scenario of several hours and hundreds of aircraft was completed in simulation.

VI. Supporting Simulation And Integration Capabilities

AS mentioned previously, we used simulations of airport surface and terminal airspace traffic in order to provide a proof-of-concept for PROCAST. We used the NASA airport surface traffic simulation platform called Surface Operations Simulator and Scheduler (SOSS) for this purpose, also. In order to enable seamless simulation of aircraft movement from gate to metroplex terminal airspace boundary, we made some enhancements to the SOSS simulation platform. Appendix A provides a brief background on SOSS and its capabilities, and then describes the capability-enhancements that we made to SOSS.

Moreover, generating the benefits analysis test-bed involved integration of multiple sub-components of PROCAST with each other and with the SOSS simulation platform. Integration of different sub-components of PROCAST (i.e., individual software modules such as NACRE and BBNs) and simulation platform, produced by different sub-project teams into a single, unified system, was a challenging task. There were three different development efforts progressing in parallel during the middle-phase of the project—(i) developing a MATLAB-based framework that included BBN-based prediction capability, (ii) enhancing NASA's SOSS simulation platform for extending its simulation capabilities to cover the terminal airspace, and (iii) developing the NACRE genetic algorithm for optimizing 4D trajectories. These three development processes were interdependent and were being developed by geographically distributed teams. The modeling requirements for all processes usually changed based on the problems faced in the development of a single process and modeling decisions made for resolving those problems. Our team adopted an Agile integration process for ensuring timely, correct, and seamless integration of these sub-components, which were being developed in parallel by geographically distributed sub-teams, as well as for managing changing modeling requirements. The Agile software integration process provided us with useful and timely feedback on correctness and integration issues of the different components throughout development, and helped reduce the development time. Appendix B describes this Agile integration process.

VII. Proof-of-concept Simulation Experiment Design

ONE of the aims of this project was to provide a preliminary proof-of-concept of the PROCAST DST as applied to the management of traffic in congested regions of the NAS—especially, busy metropolitan areas with multiple busy airports in close proximity of each other. As the first step towards the eventual evaluation in a full multi-airport traffic scenario, in Year 1 research work, we focused on modeling, simulating, and evaluating the performance of PROCAST as applied to the management of air traffic on the airport surface and in the terminal airspace at a single airport. We selected JFK as the test site because of the presence of significant arrival and departure delays as well as multiple airspace and surface arrival-departure interactions present at that airport in today's operations. The proof-of-concept validation approach was to compare the performance of PROCAST against baseline operations that represent traffic under current-day traffic management procedures. In other words, we simulate the JFK surface and terminal airspace traffic as controlled by current-day decision support tools and procedures; we separately simulate the same traffic as controlled by futuristic, PROCAST-based decision support tools; and then compare the metrics from the two simulations to quantify the benefits provided by PROCAST.

First, we describe the current-day baseline system model used in our proof-of-concept experiments.

A. Current-day Baseline Model used in Proof-of-concept Simulations

In today's operations, especially during busy traffic periods, departures at JFK are metered by a Departure Metering System (DMS), which is operated by the Port Authority of New York and New Jersey (PANYNJ). The DMS system computes recommended target movement area entry times (TMATs) for individual departure flights in order to keep movement area taxi times and departure queue lengths manageably small, and sends these TMAT times to the airlines and aircraft operators. The airlines and aircraft operators then manage gate pushback times of individual departure flights in order to adhere to these TMAT times within a certain prescribed error window. In order to compute the TMAT times, the DMS uses proprietary algorithms that perform ration-by-schedule allocation of departure runway time-slots to departure flights while maintaining equity among the multiple airlines vying for the utilization of the departure capacity of the airport. Moreover, arrivals landing at JFK are independently metered by a time-based flow management tool called Traffic Management Advisor (TMA). TMA is a decision support tool (DST) that assists the Center Traffic Management Coordinators (TMCs) and center controllers with planning and time-based metering of major-airport arrival traffic flows. TMA enables orderly and metered-to-TRACON-capacity delivery of arrival traffic over the arrival-fixes (i.e., for entry into the terminal airspace). TMA'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. TMA's DP can handle up to five different airports within a TRACON and treats them as separate entities from the scheduling perspective.

Our current-day baseline model included emulations of the two metering tools mentioned above—the DMS and TMA.

1. DMS Emulation

In our proof-of-concept baseline simulations, we used an external departure scheduler that communicated with the SOSS simulation platform. This external departure scheduler used emulations of the JFK DMS algorithms for managing departure traffic. The DMS algorithm emulation within this external scheduler first separated out departure flights into 15-minute time-bins as measured by their *estimated times of arrival* at the runway departure queue. Each 15-minute time-bin was divided into uniformly sized time-slots equal in number to the 15-minute departure capacity of the runway system. The time-slots within each 15-minute time-bin were then allocated to different airlines in proportion to their *scheduled traffic counts* in the respective time-bin (also accounting for backlog from prior time-bins). Individual flights were then allocated to the available runway time-slots in a ration-by-schedule manner (i.e., in the same order as the *estimated times of arrival* at the runway), while maintaining the runway loading within the capacity. Once all flights were assigned slots at the runway, the movement area entry times were back-computed by subtracting the average taxi-out time from the runway time (or time-slot). These times were communicated to the airlines as target movement area entry times. Airlines were then responsible for controlling the gate pushbacks in order to adhere to the target movement area entry times within a certain error window. In our model, we further back-computed the gate pushback time from the movement area entry time, and this time became the target gate pushback time. This feature was used to emulate the airline behavior of delaying flights at the gates in order to adhere to the target movement area entry times. Airlines were allowed to manually report any airline delays if they recognized that some of their flights would not be ready in time to make their scheduled gate pushback time.

In our simulation of the baseline system, we assumed that airlines will not share information on the state of their flights' pre-pushback processes. DMS emulation assumed that the flights will pushback at their *scheduled gate departure times*, except in the case where an airline reports a pre-pushback delay. We modeled airline reporting of pre-pushback delays as in the real, operational DMS. We imposed simple rules for simulating these airline-reported delays as follows. First, the pre-pushback process model, explained above (see Section IV-B), determines and updates a flight's *pushback readiness time* periodically, which may cause *pushback readiness delays* (i.e., *pushback readiness time* minus *scheduled gate departure time*). Airlines can report a gate-delay for a flight only if the flight's *scheduled gate departure time* is less than 15 minutes in the future, and if its *pushback readiness delay* is bigger than 15 minutes. If an airline reports a *pushback readiness delay* for its flight, the *estimated runway arrival time* for that flight in the DMS calculations will be updated based on the delay reported. Otherwise, the simulated DMS will compute the flight's *estimated runway arrival time* by assuming that the flight will pushback from its gate at its *scheduled gate departure time*. In other words, in this baseline the DMS had very little knowledge of the state of the pre-pushback process—it was only given an update in the case of a large change in estimated *pushback readiness time*. Moreover, for generating *estimated times of arrival* at the runway queue (which is a key variable used in the DMS scheduling algorithms) trajectories were forecast assuming that the flights will taxi at their nominal speeds, unimpeded.

2. TMA Emulation

In our proof-of-concept baseline simulations, we used an external arrival scheduler (in addition to the external departure scheduler explained above) that communicates with the SOSS simulation platform. This external scheduler contains MATLAB code, which emulates the scheduling operations performed by TMA's DP algorithm, as explained next.

Operationally, TMA is really a part of a suite of DSTs developed by NASA called the Center-TRACON Automation System (CTAS). A CTAS component called the Trajectory Synthesizer (TS) predicts the Estimated Times of Arrival (ETAs) 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. In our TMA emulation, these reference point ETA predictions were obtained from the SOSS simulation platform. Trajectory forecasts for ETA prediction assumed that the flights will fly at their nominal speeds. As in the original TMA, our emulation of TMA's DP algorithm used these ETAs to compute de-conflicted Scheduled Times of Arrival (STAs) at the meter fix and the runway threshold. TMA-DP emulation first computed STAs to the meter-fix for each arrival-stream, while retaining the first come, first served (FCFS) order within the traffic stream (these similar traffic streams are called *stream-*

classes). TMA-DP emulation may have delayed 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 were then used to generate ETAs at the runway. Runway STAs were 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 were chosen for determining their runway STAs was computed by an *Order of Consideration* (OOC) algorithm. For more details on the OOC algorithm, please see [W00]. If the delay assigned to a particular aircraft was bigger than the delay absorption capacity (user-specifiable parameter) of the TRACON, then the excess delay was fed back to the Center, and ETAs to the meter fix were updated accordingly. The TRACON delay absorption limit is a user-specifiable parameter of TMA's 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.

3. *Combined DMS, TMA Emulation*

The operational assumption we made when simulating the baseline operations was that aircraft will adhere to arrival-fix STAs (i.e., *required arrival-fix crossing times*) and runway landing STAs (i.e., *required runway landing times*) prescribed by the TMA emulation within a certain error-window. In addition, we assumed that ramp controllers will adhere to the *required gate pushback times* prescribed by the DMS emulation within a certain error-window.

B. PROCAST Model and Variants used in Proof-of-concept Simulations

The PROCAST DST consists of two main parts—one, the BBNs for developing realistic predictions of future aircraft trajectories given the current state and historical "learned" trends; and two, the NACRE genetic algorithm for optimizing and de-conflicting the arrival-departure trajectories. In addition to the full PROCAST model consisting of the two components—BBNs and NACRE—we also assessed two variants of PROCAST: one, a BBNs-only variant which consisted of BBNs for generating multiple potential futures and used baseline scheduling algorithms (DMS and TMA emulations) instead of NACRE to perform scheduling over each individual future; and two, a NACRE-only variant which used NACRE to optimize over only one future scenario that was generated using nominal forecasts of future trajectories (i.e., BBNs were not used to compute multiple realistic futures). Next, we explain the details of each of these three assessment options—Full PROCAST, BBNs-only, and NACRE-only.

1. *Full PROCAST Model used in Proof-of-concept Simulations*

The full PROCAST model used both BBNs and NACRE, and integrated them per the PROCAST philosophy—BBNs (trained on historical data) generated multiple potential future trajectory forecasts/scenarios; then, using NACRE, it performed trajectory optimization and de-confliction on each one of those future scenarios/forecasts. Once optimization/de-confliction over a large number of scenarios is completed, probabilistic analysis was used to compute the "statistically-best" set of optimal control actions (i.e., flight release times), and this set of actions was chosen for implementation on the traffic.

We made certain operational assumptions when simulating the PROCAST operations. First, we assumed that airlines and aircraft operators will provide PROCAST with periodic estimates of *pushback readiness times* for departure flights, which have their *scheduled gate departure times* in the near future. These *pushback readiness time* updates are provided irrespective of whether the departure flight is currently waiting at its gate undergoing pre-pushback preparation (e.g., de-boarding, fuelling or boarding) or whether the prior connecting arrival flight has not even reached its gate yet and is still in its airborne or taxi-in phase. It was assumed that in the future with the availability of data-sharing technologies such as SWIM and operational procedure changes under the Collaborative Decision Making (CDM) paradigm, these data can be generated and shared between airlines and ANSPs. Second, the BBNs used these *pushback readiness time* estimates to periodically update their future trajectory forecasts. Instead of assuming that flights will taxi and/or fly at their nominal speeds, unimpeded, for trajectory forecasting, BBNs relied on historical trends for computing transit times along segments of the taxiway system depending on current (measured/estimated) values of key factors that influence the transit through these regions. For example, transit time of a departure flight from spot to departure runway depends heavily on the length of the departure queue observed at the instant the flight crossed its spot. Third, it was assumed NACRE will periodically optimize and de-conflict each of the thousands of future trajectory scenarios generated by the BBNs. In doing so, NACRE will perform combined arrival and departure trajectory optimization covering the entire flight-domain from the gate to the terminal airspace boundary. Following NACRE-optimization, PROCAST will perform statistical assessment to compute the “statistically-best” flight release times for key route-nodes: gates and runways for departures; arrival-fixes and runways for arrivals (i.e., *required gate pushback times*, *required runway takeoff times*, *required arrival-fix crossing times* and *required runway landing times*). Fourth, we assumed that PROCAST will periodically distribute these flight release times to the airlines/controllers and the airlines/controllers will convey the times to the aircraft to enforce adherence to these times within certain error-windows. Table 2 summarizes the main characteristics of the PROCAST operations model and compares it against the Baseline operations model used in our simulation-based proof-of-concept demonstration experiment.

	Baseline Operations	PROCAST Operations
Scheduling method for ***Departures***	• Simple, deterministic departures-only planning similar to current-day Departure Management Tools • Uses nominal pushback readiness time estimates	• Combined arrival-departure planning • Assumes periodic updates of pre-pushback process state • BBNs generate multiple futures • Including estimates of pushback readiness times and times of arrival at key nodes in the surface-terminal network • GA optimizes arrival and departure operations over each future • Statistical assessment selects best flight release times
Scheduling method for ***Arrivals***	Deterministic arrivals-only planning based on current-day Traffic Management Advisor (TMA) scheduling algorithms	

Table 2. Comparison of baseline and PROCAST operations for simulation-based proof-of-concept demonstration.

2. *PROCAST Variant #1: BBNs-only Operations*

This variant only leveraged the BBNs part of PROCAST. Under BBNs-only operations, as in the full PROCAST operations, we assumed that airlines and aircraft operators will provide the planning function with periodic estimates of *pushback readiness times* for departure flights, which have their *scheduled gate departure times* in the near future. BBNs use these estimates to periodically update their future trajectory forecasts. In contrast to full PROCAST, however, NACRE was not used for optimizing and de-conflicting over each of these future trajectory forecasts. Instead, baseline scheduling methods (i.e., DMS and TMA emulations), which perform departure scheduling independent of arrival scheduling, were used to schedule and de-conflict each of the trajectory forecast scenarios. Following the baseline scheduling over multiple future scenarios, statistical assessment was performed to compute the “statistically-best” flight release times for key route-nodes. These flight release times ware periodically distributed to the airlines/controllers (in our experiments, the flight release times were really communicated to SOSS, which contains models of airline and controller actions).

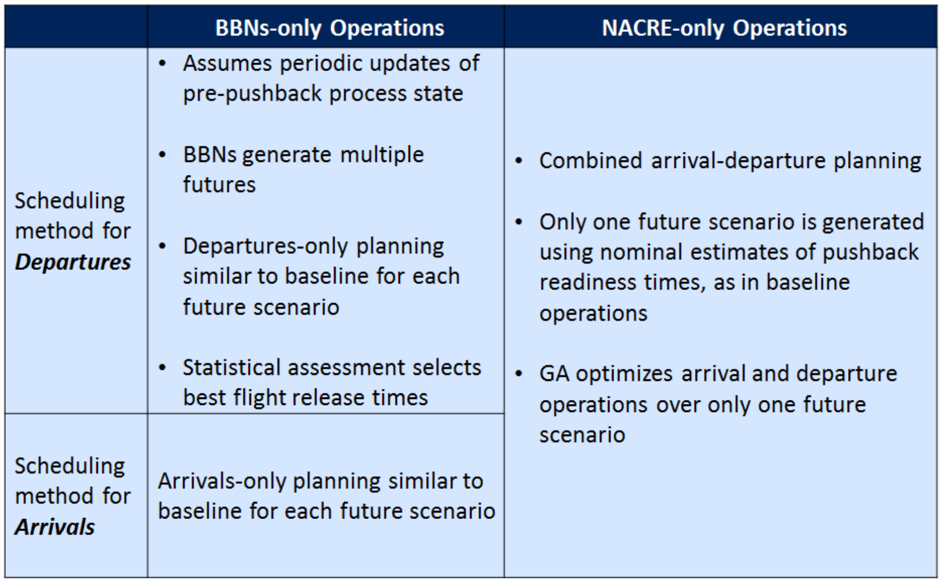
The purpose of studying this variant of PROCAST was to find out the delay-saving/throughput-maximizing/other benefit-causing impact of BBNs only, without the added beneficial effect of NACRE’s 4D trajectory *optimization*.

3. *PROCAST Variant #2: NACRE-only Operations*

This variant only leverages the NACRE part of PROCAST. Under the NACRE-only operations, we assumed that airlines will not share information on the state of their flights’ pre-pushback processes. The NACRE-only planning function assumed that the flights will depart their gates at their scheduled gate departure time, except in the case where an airline notifies the planning function about a pre-pushback delay, as *explained* in Section VII-A-1. If the airline notifies the planning function of a pre-pushback delay, then the delayed *pushback readiness time* is also provided to the planning function. In this variant, BBNs were not used for generating thousands of realistic future trajectory forecasts. Only one future traffic scenario was generated using the nominal trajectory forecast, similar to the baseline operations scenario (i.e., trajectories were forecast assuming that the flight will taxi and/or fly at their nominal speeds). The NACRE genetic algorithm optimized and de-conflicted the trajectories starting with this one initial trajectory forecast. The flight release times computed by NACRE were periodically distributed to the airlines/controllers (in our experiments, the flight release times were really communicated to SOSS, which contains models of airline and controller actions).

The purpose of assessing this variant was to find out the delay-saving, throughput-maximizing, other benefit-*causing* impact of NACRE only, without the added beneficial effect of stochastic evaluation offered by the BBNs.

Table 3 summarizes the main characteristics of the two PROCAST variants used in our simulation-based proof-of-concept demonstration experiment.



	BBNs-only Operations	NACRE-only Operations
Scheduling method for ***Departures***	• Assumes periodic updates of pre-pushback process state • BBNs generate multiple futures • Departures-only planning similar to baseline for each future scenario • Statistical assessment selects best flight release times	• Combined arrival-departure planning • Only one future scenario is generated using nominal estimates of pushback readiness times, as in baseline operations • GA optimizes arrival and departure operations over only one future scenario
Scheduling method for ***Arrivals***	Arrivals-only planning similar to baseline for each future scenario	

Table 3. Comparison of BBNs-only and NACRE-only variants of PROCAST.

C. Traffic Scenarios used for Simulation-based Assessments

As mentioned previously, we selected JFK as the focus airport for our study and used NASA’s SOSS for our simulation experiments. SOSS is a fast-time simulation platform used to simulate airport surface operations and support rapid prototyping of surface scheduling algorithms. For generating realistic SOSS input data (runway configurations, traffic datasets, etc.), we first conducted a study of meteorological conditions and active runway configurations at the JFK airport in order to select a frequently used runway configuration. Based on our analysis, a commonly used runway configuration at JFK is runways 31L and 31R for arrivals and runway 31L for departures (see Figure 8).

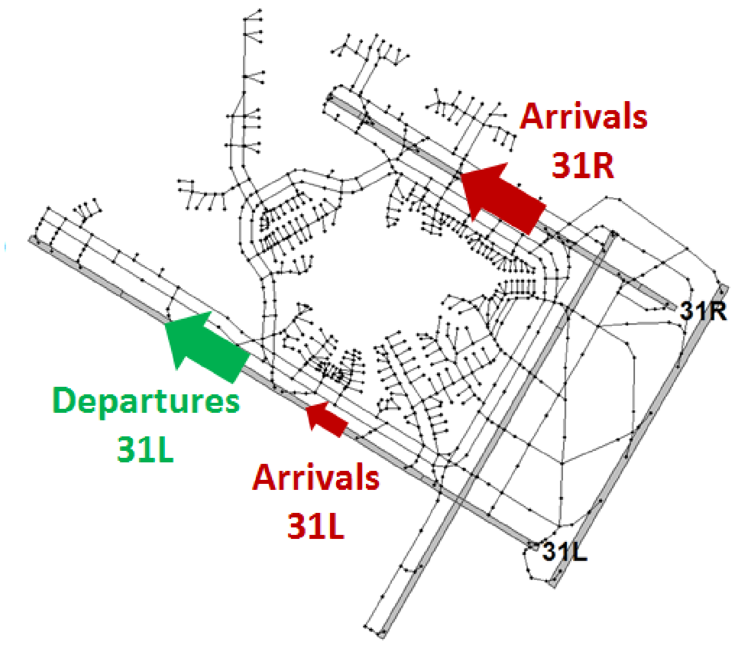


Figure 8. JFK runway configuration selected for our simulation-based studies.

We generated three simulation traffic scenarios for JFK, each two hours in length, using a Saab proprietary detailed process of analyzing and characterizing recorded ASDE-X, ASQP, and ASDI data from busy time-periods on three different days in year 2009. Each of these traffic scenarios consisted of scheduled flight times (*scheduled gate departure times* and *scheduled arrival-fix crossing times*), gate/runway allocations, and taxi routes for the flights involved in the scenarios. Table 4 provides details of the three selected traffic scenarios.

Traffic Scenario Day/Time	Number of Departures	Number of Arrivals
November 24, 7-9 PM Local Time	82	63
November 27, 8-10 PM Local Time	72	60
October 21, 11 AM-1 PM Local Time	62	90

Table 4. JFK traffic scenarios selected for simulation-based assessments.

D. Method of Simulation-based Assessment

Fast-time simulation-based assessments were conducted using NASA's SOSS simulation platform to test and compare operations under the control of four decision support tool configurations—current-day baseline, full PROCAST, BBNs-only variant of PROCAST and NACRE-only variant of PROCAST. Simulations for each of the decision support tool configurations was performed for each of the three selected two-hour traffic scenarios. With the exception of the choice of the scheduler (i.e., the traffic management/planning function), all other settings were consistent between the simulations. Specifically, the *scheduler call interval* and *forecast window* parameters were set at 300 seconds and 15 minutes, respectively. The *scheduler call interval* defines how often the scheduler is invoked by SOSS. The *forecast window* defines how far in the future the scheduler forecasts flight trajectories each time it is invoked.

Each of the schedulers computed *required gate release times* for departures as well as *required arrival-fix crossing times* for arrivals, and sent them back to SOSS periodically (once every *scheduler call interval*). SOSS adhered to these *Scheduled Times of Release* (STRs) from respective nodes in the flights' routes. In addition to the delays imposed by adherence to STRs, SOSS also imposed delays (speed reductions or complete stops) to avoid physical conflicts between flights on the airport surface as well as to satisfy the standard FAA-defined aircraft weight-class-based separations between consecutive runway operations. SOSS imposed these separation-maintaining delays using its internal CD&R module.

Uncertainty in pre-pushback processes for departures was simulated using the model explained in Section . This stochastic model was used to compute the *actual gate pushback readiness time* for each departure flight (that has not yet pushed back) at each scheduler call time. The actual gate pushback readiness times were not available to any of the schedulers—very rough estimates (assumes that the flights will be ready for pushback at their scheduled gate departure times, except if the airline reports any pre-pushback delays) were available to the baseline and NACRE-only schedulers; intelligent BBN model-based estimates were available to the full PROCAST and BBNs-only schedulers.

After performing all the simulations, the benefits provided over baseline operations by the full PROCAST scheduler and its two variants were assessed by comparing performance metrics, such as delay, taxi time, and throughput. The next section provides details of the benefits-assessment results.

VIII. Phase I Simulation Results

THE aim of the Phase I simulation experiments was to provide a PROCAST proof-of-concept by quantifying the benefits provided over the current-day baseline scheduler by the full PROCAST scheduler and its two variants. These benefits were quantified by assessing a number of performance metrics. For departure flights, many performance metrics (especially delay) were divided into four parts, depending upon location—gate, ramp area, movement area, and terminal airspace. For example departure delays were segregated into gate delay, ramp area taxi delay, movement area taxi delay, and airborne delay. Gate delay is computed as the difference between the *actual gate pushback readiness time* and *actual gate release time*—this represents the time for which the scheduler holds the aircraft back at the gate beyond the time when it is ready for pushback. *Ramp area taxi delay* is computed as the difference between *unimpeded ramp taxi time* and *actual ramp taxi time*. *Movement area taxi delay* is computed as the difference between *unimpeded movement area taxi time* and *actual movement area taxi time*. *Movement area taxi time* is computed as the difference between *spot release time* and *runway release time*. Any time spent in the runway departure queue is included in this metric. Airborne delay is the difference between the actual and nominal *runway-to-departure fix transit times*. Other performance metrics such as throughput were measured at a specified single location only, e.g., *runway departure throughput*. Similarly, for arrivals some of the performance metrics were divided into three parts—airborne, movement area and ramp area (e.g., *airborne arrival delay, movement area taxi delay and ramp area taxi delay)*. Whereas, other metrics such as throughput were computed at specific points, e.g., *runway arrival throughput*.

Figure 9 shows the distribution of delay for the four test configurations (baseline, BBNs-only, NACRE-only, and full PROCAST) when applied to the traffic scenario from November 24, 2013 7-9 PM Local Time. Delays shown here are total delays summed up over the 82 departures in the traffic scenario. We do not show arrival flight delays here because these delays were roughly the same across the four scheduler configurations. Note that the pre-pushback delays (i.e., delays simulated by our randomized pre-pushback process model) are the same across the four configurations. This is because, although computation of the pre-pushback delays (i.e., computation of randomized pushback readiness times) is random, the same random numbers are stored and applied to departure flights across the four configuration simulation runs to enable fair apples-to-apples comparison of delay and other metrics. Furthermore, we only show the total taxi delay—not broken down into ramp and movement area taxi delays. Also, airborne delay is not shown here because it was a very small part of the total delay and almost the same across the four scheduler configurations.

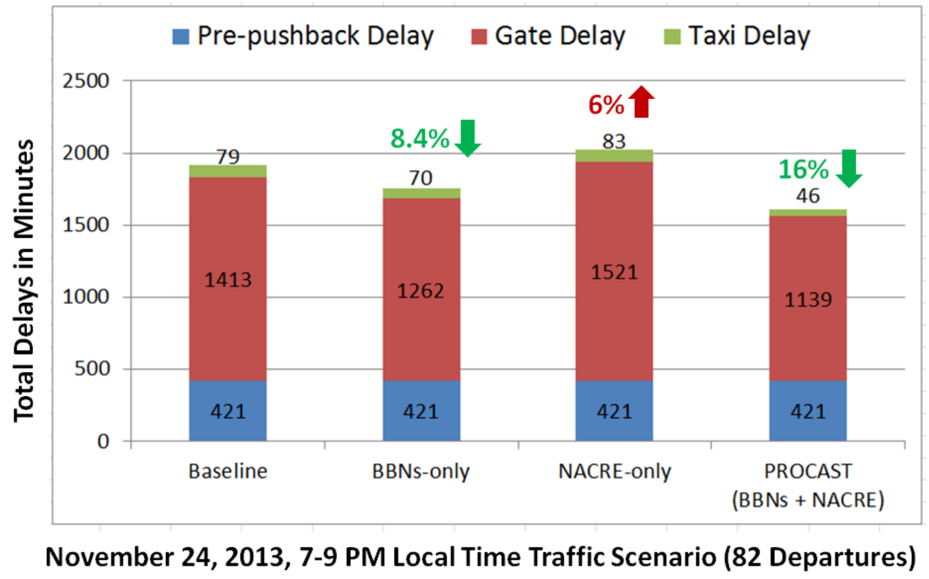


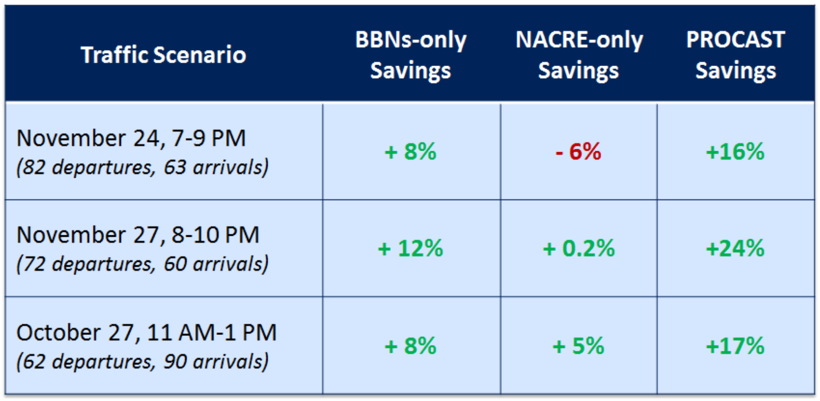
Figure 9. Delay distribution over departure flights for the four test configurations (for November 24, 2013 7-9 PM local time traffic scenario).

As shown in the figure, the BBNs-only scheduler reduced the total delay (pre-pushback delay plus gate delay plus taxi delay) by around 8% as compared to the baseline scheduler for this traffic scenario. The reduction in delay is mainly because the BBNs-only scheduler used information about the measured pre-pushback state of departure flights more effectively in order to estimate and manage demand for the departure runway. The baseline scheduler, on the other hand, did not do a good job at managing departure runway demand because it worked off of erroneous pushback readiness time estimates for many departures based on the assumption that most of the departure flights will be ready to pushback at their published *scheduled gate departure times*.

Figure 9 also shows that the NACRE-only scheduler, in fact, managed the traffic with around 6% more total delay as compared to the baseline operations. This was because NACRE, although it involves runway sequence optimization and taxiway conflict detection and resolution, (just like the baseline scheduler) worked off of erroneous pushback readiness time estimates for many departures based on the assumption that most of the departure flights will be ready to pushback at their published *scheduled gate departure times*. Performing sequence optimization based on erroneous *pushback readiness time* estimates (leading to erroneous *runway queue entry time* estimates) caused the performance to degrade below the baseline operations delay levels. It can be seen from the figure that the main manifestation of runway sequence planning using erroneous *runway queue entry time* estimates was a large increase in gate delays—many departures were held back at their gates in order to let other departures go ahead of them and utilize available runway time-slots, when in fact (unknown to the NACRE-only scheduler) these other departures were experiencing pre-pushback delays (i.e., were not ready to pushback and could not utilize those runway time-slots). Hence, runway departure capacity went unused while departures which were held back at the gates by the NACRE-only scheduler could have utilized that capacity.

When BBNs were integrated with NACRE to form the full PROCAST scheduler and applied to the November 24, 2013 traffic scenario, Figure 9 shows that this saved around 16% of the total delay as compared to the baseline scheduler. There were two reasons for this big delay savings—one, the scheduler used information about the measured pre-pushback state of departure flights more effectively in order to estimate and manage demand for the departure runway; and two, the scheduler performed runway sequence optimization and conflict detection & resolution with more accurate/informed estimates of *runway departure queue entry times* and *taxiway intersection-point crossing times*.

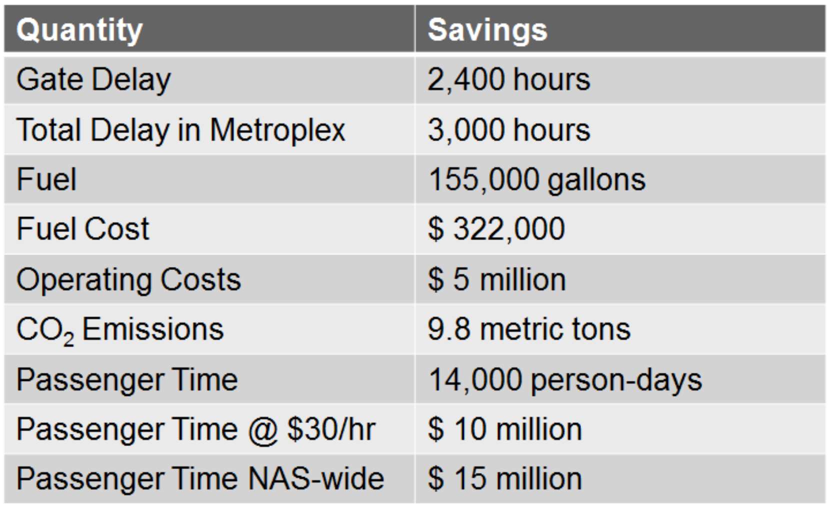
Up to this point, we have shown and discussed the delay metrics for only one traffic scenario (November 24, 7-9 PM Local Time). We repeated the simulations over the two other traffic scenarios that we had generated (see Table 4). Table 5 summarizes the departure delay saving benefits provided by the full PROCAST scheduler and its two variants over the baseline scheduler. As seen from Table 5, the delay savings over the three scenarios follow a similar trend—the BBNs-only scheduler provides significant delay savings of around 8-12% depending upon the scenario; the NACRE-only scheduler provides only minor or negative benefits because it involves runway sequence optimization using potentially erroneous *runway queue entry time* estimates; finally, the full PROCAST scheduler provides much larger (as compared to BBNs-only scheduler) benefits of around 16-24% depending upon the traffic scenario.



Traffic Scenario	BBNs-only Savings	NACRE-only Savings	PROCAST Savings
November 24, 7-9 PM *(82 departures, 63 arrivals)*	+ 8%	- 6%	+16%
November 27, 8-10 PM *(72 departures, 60 arrivals)*	+ 12%	+ 0.2%	+24%
October 27, 11 AM-1 PM *(62 departures, 90 arrivals)*	+ 8%	+ 5%	+17%

Table 5. Departure delay saving benefits over multiple traffic scenarios.

In order to provide a broader perspective on the potential for performance benefits provided by PROCAST, we took an average of multiple different metrics (delays, taxi times, airborne transit times, etc.) over the three traffic scenarios for the PROCAST operations and the baseline operations, and converted these metrics to annualized potential savings metrics, as shown in Table 6 below.



Quantity	Savings
Gate Delay	2,400 hours
Total Delay in Metroplex	3,000 hours
Fuel	155,000 gallons
Fuel Cost	$ 322,000
Operating Costs	$ 5 million
CO_2 Emissions	9.8 metric tons
Passenger Time	14,000 person-days
Passenger Time @ $30/hr	$ 10 million
Passenger Time NAS-wide	$ 15 million

Table 6. PROCAST estimated annual savings as compared to a current-day baseline.

We used the following assumptions while computing these annualized savings estimates:

- Conditions similar to the three traffic scenarios we studied prevail for two hours on 100 days per year
- *Fuel burn rate = 8 kg / min taxiing, 40 kg / min airborne, cost = $ 993.60 / metric ton*
- *Operating costs = $ 27 / min at gate, $ 41 / min taxiing, $ 78 / min airborne*
- 1 minute savings in NYC = 1.5 minute savings NAS-wide

These assumptions have been used by Saab [SL13] and others in past benefit assessment studies of surface traffic management tools, and have been validated by the surface benefits assessment research community. Table 6 shows that PROCAST has a significant potential to provide delay, fuel, operating cost, emissions, and passenger time savings. Based on our past benefits assessment work, we estimate that if PROCAST were extended to manage the traffic in the entire New York metroplex, its benefits could be at least five times bigger than the Table 6 estimates, which assume the application of PROCAST at JFK Airport only.

A. Limitations

This sections outlines some of the limitations associated with the current state of the prediction and optimization tools that we have developed in Phase I. These limitations serve as lessons learned and will motivate corrective actions in Phase II research.

1. *Airlines’ Economic Objectives Not Directly Modeled in Optimization Cost Function*

The cost function considered in the Phase I version of the NACRE 4D trajectory optimization was a weighted sum of squares of individual aircraft delays. We chose to use a sum of squares instead of a linear sum because we wanted to avoid situations where the optimization will save delays for many aircraft by unfairly allocating large delays to a small number of aircraft. However, the cost function did not involve consideration of the complex factors that airlines assess when making prioritization decisions between their departure/arrival flights. These factors include trade-offs between gate delays and fuel consumption, early landing time versus excess taxi-in delay, fast TRACON descent versus fuel-optimal descent and other factors. In Phase II research, we plan to remedy this limitation by considering a more comprehensive objective function that characterizes the airlines’ economic objectives more precisely.

2. *Conflict Detection & Resolution (CD&R) Logic Not Directly Integrated into Trajectory Optimization*

In Phase I work, NACRE consisted of a two-step optimization—step 1 computed an optimal sequence of runway utilization between arrival and departure flights; step 2 worked back from this optimized sequence and tried to resolve any conflicts that may occur on the taxiway system by holding flights slightly longer or shorter than required at their gates. No CD&R delays were applied to the flights on the taxiway system. This sometimes unfairly favored flights that had left their gates over conflicting flights that were currently waiting at their gates. In Phase II, we plan to enhance the trajectory optimization algorithm by integrating CD&R function directly into optimization.

3. *Airline Equity Considerations Not Considered in NACRE Optimization*

The Phase I NACRE optimization function does not have the logic to enable equitable allocation of available departure runway capacity among different airlines. As explained above, our baseline model of departure traffic management does include this capability. It rations the available departure capacity among airlines based on their scheduled departures in each 15-minute time-bin (and their backlog from previous time-bins). If a particular airline does not have enough time-slots allocated to it in a particular time-bin to fill its departures, then it will have to hold those departures at the gate (or in the ramp area) until the next time-bin. The purpose of this logic is to avoid giving unfair advantage to one airline over the other, and this commitment to fairness has been crucial in securing airlines’ approval for participating in departure metering programs such as the one currently run at the JFK Airport. The NACRE optimization function in its current form allows an airline to unfairly exceed its allocated quota. This might be a reason why we see large delay saving benefits for the case of the full PROCAST scheduler (which contains both BBNs as well as NACRE). The BBNs-only scheduler, however, uses a ration-by-schedule metering algorithm for departure metering, and this algorithm ensures that no airline gets runway departure slots beyond its scheduled quota to ensure fairness. The delay savings benefits predicted for the BBNs-only scheduler are, therefore, a more realistic estimate of the potential benefits of PROCAST.

Further, a comparison of the runway utilization sequences computed by the genetic optimization algorithm in NACRE against first-come first-served runway utilization sequences did not show any significant differences. As a result, we postulate that the incremental delay savings provided by the full PROCAST scheduler over the BBNs-only scheduler may not be because of any runway utilization sequence optimization but mainly because the NACRE optimization in the full PROCAST scheduler neglected fairness constraints and hence was able to depart more aircraft from particular airlines at unfair disadvantage to other airlines. (Note: the difference between these two schedulers is that full PROCAST scheduler contains genetic optimization scheduling while BBNs-only scheduler contains ration-by-scheduling.)

In Phase II research, we address this issue by modifying the NACRE optimization function (or potentially replacing it with another optimization algorithm) such that it maintains equity among airlines.

4. Computational Limitations Restrict Number of Potential Futures Used for Evaluation

Our original intent was to conduct assessment by evaluating optimizations performed over thousands of different trajectory forecasts. However, due to computational limitations (we used a standard desktop system), we had to restrict ourselves to evaluating over only around 50 potential futures in the case of the BBNs-only and full PROCAST schedulers. Comparison of total delay numbers for 1, 5, 10, 50 potential futures shows that the delays reduce with increasing number of potential future evaluations. Hence, we believe that there would be a big benefit in increasing the number of potential futures that we assess over. Phase II research will assess computation speed enhancement strategies for enabling real-time evaluation of thousands of potential futures.

IX. Preliminary PROCAST Concept of Operations

IN addition to simulation-based proof-of-concept demonstration, we have also developed a preliminary concept of operations (ConOps) for an integrated arrival-departure-surface traffic management decision support tool for improved management of air traffic in the entire New York metroplex. In Phase II of this project, we aim to extend the prediction and scheduling capabilities of the PROCAST tool and develop a prototype decision support tool that follows the ConOps that we describe in this section. Of course, as the work progresses and as we hold discussions with different stakeholders (including eventual users of this tool), the ConOps is subject to change based on feedback.

A. Motivation for PROCAST-based Integrated Arrival-Departure-Surface Traffic Management

Significant operational and environmental consequences exist when the demand for air traffic services exceeds the capacity of the system. At the airport surface, the imbalance between demand and capacity manifests in large numbers of aircraft on the taxiways waiting in line for air traffic control (ATC) clearance to take off. During periods of taxiway congestion, aircraft are burning fuel unnecessarily, which generates environmentally harmful emissions and reduces the operational efficiency of the airport.

One cause of congestion is that aircraft operators (i.e., airlines and business/private aircraft operators) have their aircraft push back from the gate or leave the parking area based on a schedule which is independent of any other consideration. Operational gate/ramp personnel are rewarded for ensuring that the aircraft under their control enter the taxiway network “on schedule” or as promised to customers.

Moreover, at airports like JFK where a departure metering system exists today, the planning, which involves allocating departure runway time-slots to individual aircraft, may be thrown off track by large uncertainties in the pre-pushback processes. Due to these large uncertainties, estimates of pushback readiness times of individual departure flights could be in error by more than +/- 15 minutes frequently. The departure metering system relies on these pushback readiness time estimates for computing departure runway demand. It usually works in time-discretization of 15-minute time-bins for first distributing departure runway capacity among airlines and then allocating time-slots to the airlines’ individual departures. Hence, departure runway slot allocation based on erroneous pushback readiness time estimates may cause inefficient operations (as we saw in the baseline scheduler and NACRE-only scheduler simulation results).

After pushback, entry to the taxiways is under the control of the air traffic controllers whose primary mission is safe maneuvering of the aircraft in the taxiways to the runway; the region of runways and taxiways under ATC control is referred to the movement area. Given that the controllers are presented with a narrow time-window within which to plan and deliver clearances, the controllers attempt to re-order the relatively random order in which departures are scheduled to enter the movement area across the airport as a whole. As the result, the ground controllers make tactical decisions to issue clearances to maintain safety and runway throughput. The sequence of departures delivered to the runway may be set by the entry order to the movement area, the skill of the ground controllers, traffic management initiatives and constraints, and the physical limitations of the taxiway structure. Under these circumstances, air traffic controllers attempt to create the “best” departure queue sequence of aircraft to the runway in order to achieve the highest departure rate after the aircraft have entered the movement area, while not sacrificing safety. But, constraints in the airspace (which are especially restrictive in busy, interconnected airspaces such as the New York TRACON) are usually not considered when making runway sequence decisions, except through application of coarse miles-in-trail restrictions or rough ordering to avoid consecutive flights to the same departure-fix. As a result, surface-only planning focus may result in inefficient overall metroplex (surface plus airspace) operations.

B. Proposed PROCAST Traffic Management System

The fundamental design objectives of the PROCAST system are to: (1) integrate and enhance the operational efficiency of the metroplex airport surface and terminal airspace traffic movement by equitable distribution of capacity at multiple surface and airspace constraint points (e.g., departure/arrival runways, departure-fixes, arrival-fixes) under specified operational conditions, (2) reduce the economic and environmentally harmful effects of airport and terminal airspace congestion and delay, (3) provide operationally useful information to all relevant operational personnel, and (4) avoid impacts on the FAA workload and authority.

In order to fulfil these objectives, we have developed a preliminary PROCAST ConOps, which describes how a decision support tool (or an array of tools) based on PROCAST’s prediction and schedule-optimization capabilities can be utilized in different ATC facilities in the New York region. Figure 10 shows the expected inputs into the PROCAST system coming from different FAA and airline systems.

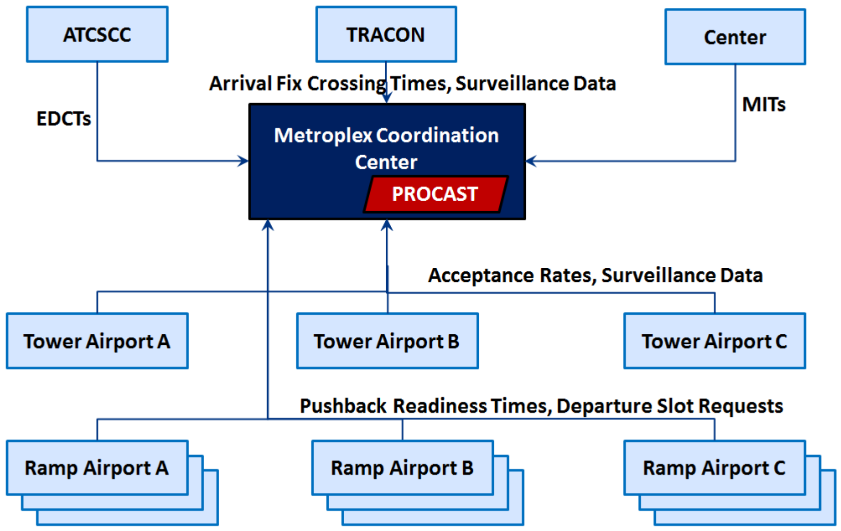

ATCSCC
TRACON
Center
Arrival Fix Crossing Times, Surveillance Data
EDCTs
MITs
Metroplex Coordination Center
PROCAST
Acceptance Rates, Surveillance Data
Tower Airport A
Tower Airport B
Tower Airport C
Pushback Readiness Times, Departure Slot Requests
Ramp Airport A
Ramp Airport B
Ramp Airport C

Figure 10. PROCAST ConOps—system inputs.

Figure 11 below shows the expected outputs (i.e., controls) provided by the PROCAST system to different FAA and airline facilities.

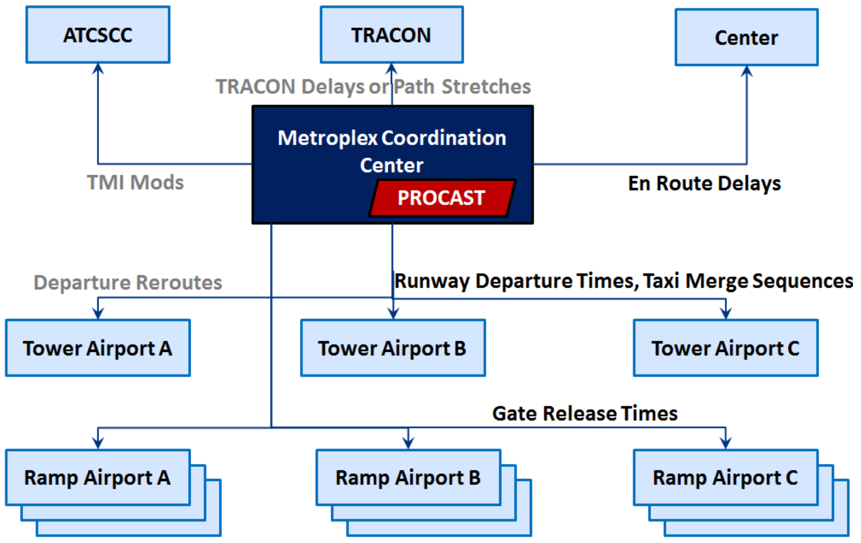


Figure 11. PROCAST ConOps—system outputs.

As shown in Figure 10, the FAA facilities—ATCSCC and Center—provide traffic flow management function. These facilities provide the following inputs to the PROCAST system (shown in red)—Expect Departure Clearance Times (EDCTs) for any active Ground Delay Programs (GDPs) that impact New York departures and miles-in-trail (MIT) restrictions that may impact the New York TRACON arrival or departure traffic. We posit the existence of a new entity, referred to as the 'Metroplex Coordination Center' (MCC), which will operate the PROCAST function and will ensure that the metroplex operates for the greater good of all participants, instead of as a separate entities operating in a conflicting condition. In addition to EDCTs and MITs, the MCC will also receive multiple inputs including (i) arrival-fix crossing time estimates from the TMA system at the local TRACON or center, (ii) acceptance rates and runway departure queue entry time estimates from the local airports, and (iii) pushback readiness time estimates, pre-pushback process state updates, and departure slot requests from airlines.

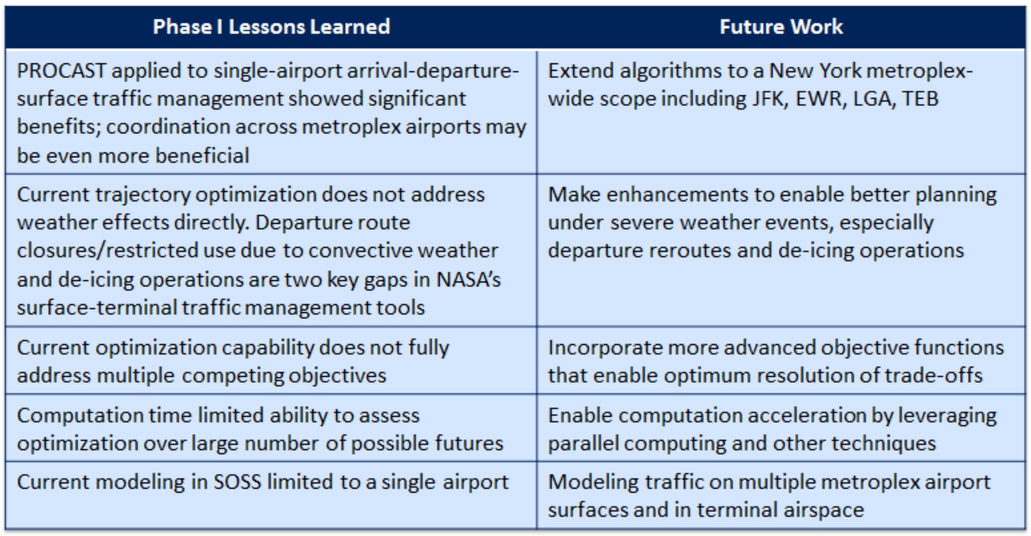
The MCC will use the PROCAST system to generate airport-specific schedules for departures in the form of target take-off times (TTOTs) while rationing departures per departure-fix. TTOTs may be communicated to the local-airport departure metering system in order to help generation of the departure sequence to the ramp and the assignment of flights to a departure fix-runway combination (as shown in Figure 11); or in absence of a departure meteting system at the local airport, the MCC may directly back-compute a Target Movement Area Entry Time (TMAT) or Target Off Block Time (TOBT) for sending to the local airport. The MCC may also use PROCAST to compute en route delays for arrival flights and communicate these delays to the TMA system in the center. Further, using taxi conflict detection and resolution logic within NACRE, the MCC may also recommend taxi merge sequences to the ground controllers at the local airports. In future versions of the PROCAST algorithm, it may be extended to provide more recommendations—for example, departure reroute assignments in the event of departure-fix closures or capacity-degradations due to weather, TRACON delay or path stretch advisories, and modifications to MIT or EDCT TMIs; these future applications are shown in grey in Figure 11.

X. Summary, Conclusions, and Future Work

MULTIPLE complex problems facing the world today are characterized by three key attributes: (i) presence of complex interactions between system components and network impacts, (ii) uncertain future system behavior, and (iii) multiple, competing, non-linear objectives. Managing highly interconnected streams of air traffic at airports and terminal airspaces in busy metropolitan areas is one such problem. This Phase I project developed an innovative, computationally-efficient framework for generating globally optimal solutions for complex system problems that are characterized by the above three attributes. We applied this approach, PROCAST, to develop a spacing-and-scheduling DST for integrated arrival-departure-surface spacing and scheduling, covering two flight domains: terminal airspace and airport surface. The innovative approach combines two technologies— (i) BBNs to generate multiple potential futures given the current state of the airport surface-terminal airspace system, and (ii) NACRE to optimize over each of these potential futures. Then, we select the statistically-best optimal control actions for implementation.

We performed preliminary proof-of-concept simulation experiment that demonstrates the viability of the PROCAST approach as applied to the control of arrival, departure and surface traffic at JFK. Simulation experiments compared JFK arrival-departure-surface operations as controlled by a model of current-day baseline operations against the same operations as controlled by PROCAST and two variants of PROCAST (a BBNs-only variant and a NACRE only variant). Simulation experiments used three realistic traffic scenarios derived from recorded surface surveillance data, each traffic scenario being two hours in length and representing a busy traffic period at JFK. We modeled realistic uncertainties in the pre-pushback processes of the departure flights as well as during the transit of the flights along the airport surface. In all the simulations, the BBNs-only and full PROCAST schedulers provided significant delay savings over the baseline operations, whereas NACRE-only scheduler did not provide much benefit (and sometimes made operations worse). Moreover, the delay savings over the three scenarios followed a similar trend—the BBNs-only scheduler provided significant delay savings of around 8-12%; the NACRE-only scheduler provided only minor or negative benefits; finally, the full PROCAST scheduler provided much larger (as compared to BBNs-only scheduler) benefits of around 16-24% depending upon the traffic scenario. When these delay and other benefit numbers are annualized, we estimate that PROCAST on an average will save around 3000 hours of total metroplex delay (ground plus airborne delay, gate plus taxi plus airborne delay), 155,000 gallons of fuel, \$5 million of airline operating cost, and 14,000 person-days of passenger time, which amounts to \$15 million in passenger delays saved NAS-wide.

Given these benefit numbers, we believe that PROCAST is a very promising technology. There were certain lessons learned and limitations associated with the Phase I version of PROCAST. Table 7 lists these limitations and outlines the potential Phase II corrections that they motivate.



Phase I Lessons Learned	Future Work
PROCAST applied to single-airport arrival-departure-surface traffic management showed significant benefits; coordination across metroplex airports may be even more beneficial	Extend algorithms to a New York metroplex-wide scope including JFK, EWR, LGA, TEB
Current trajectory optimization does not address weather effects directly. Departure route closures/restricted use due to convective weather and de-icing operations are two key gaps in NASA’s surface-terminal traffic management tools	Make enhancements to enable better planning under severe weather events, especially departure reroutes and de-icing operations
Current optimization capability does not fully address multiple competing objectives	Incorporate more advanced objective functions that enable optimum resolution of trade-offs
Computation time limited ability to assess optimization over large number of possible futures	Enable computation acceleration by leveraging parallel computing and other techniques
Current modeling in SOSS limited to a single airport	Modeling traffic on multiple metroplex airport surfaces and in terminal airspace

Table 7. Phase I Limitations/Lessons Learned And Potential Future Work

As shown in Table 7, future (Phase II) research work should focus on extending the algorithms (both predictive BBN algorithms as well as scheduling/optimization algorithms) to solve a metroplex problem where multiple proximate airports vie for the use of commonly shared airspace and fixes. Further, it should focus on enhancing the trajectory optimization capability by incorporating weather-sensitive traffic planning and advanced objective functions for optimum resolution of important trade-offs. Future work should explore parallel computing and other methods for improving the speed of computation so that we can evaluate plans over thousands of potential futures. Finally, NASA’s simulation platform needs to be enhanced to enable simulation of surface and air traffic to/from multiple metroplex airports.

Acknowledgments

The authors would like to thank NARI for the support of this project and for fostering collaboration with NASA and LEARN researchers. We would also like to thank Robert Windhorst, Yoon Jung (NASA Ames) for letting us use SOSS; Zhifan Zhu and Sergei Gridnev (NASA) for SOSS software support; NASA SARDA, ATD-2, IADS, and New York TBO researchers for positive feedback throughout the project; Kristin Rozier, Johann Schumann (NASA Ames) for pointers on Bayesian Belief Network software.

References

[AM10] Anderson, R., & Milutinovic, D. (2010). Optimization of Taxiway Traversal at Congested Airports. In *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*.

[BT12] Bernd O., Thomas K., Michael S., Hartmut F., Vivek K., Lance S., “Turnaround Prediction with Stochastic Process Times and Airport specific Delay Pattern,” International Conference on Research in Airport Transportation (ICRAT), Berkeley, 2012.

[CH12] Coppenbarger, R., Hayashi, M., Nagle, G., Sweet, D., and Salcido, R., "The Efficient Descent Advisor: Technology Validation and Transition," AIAA-2012-5611, 12th American Institute of Aeronautics and Astronautics (AIAA) Aviation Technology, Integration, and Operations (ATIO) Conference, Indianapolis, IN, 17-19 Sep. 2012

[CM07] Coppenbarger, R. A., Mead, R. W., and Sweet, D. N., "Field Evaluation of the Tailored Arrivals Concept for Datalink-Enabled Continuous Descent Approach," AIAA-2007-7778, AIAA Aviation Technology, Integration and Operations Conference (ATIO), Belfast, Northern Ireland, 18-20 Sep. 2007.

[GD03] Gotteland, J. B., & Durand, N. (2003). Genetic algorithms applied to airport ground traffic optimization. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on* (Vol. 1, pp. 544-551). IEEE.

[GM09] Gupta, G., Malik, W., & Jung, Y. C. (2009). A mixed integer linear program for airport departure scheduling. In *9th AIAA aviation technology, integration, and operations conference (ATIO)* (pp. 21-23).

[GV01] Green, S. M., and Vivona, R. A., "En route Descent Advisor Concept for Arrival Metering," AIAA-2001-4114, Guidance, Navigation, and Control Conference, Montreal, Canada, Aug. 2001.

[HD08] Hu, X. B., & Di Paolo, E. (2008). Binary-representation-based genetic algorithm for aircraft arrival sequencing and scheduling. *Intelligent Transportation Systems, IEEE Transactions on*, *9*(2), 301-310.

[IR01] Isaacson, D. R., and Robinson III, J. E., "A Knowledge-based Conflict Resolution Algorithm for Terminal Area Air Traffic Control Advisory Generation," AIAA Guidance, Navigation, and Control Conference, Montreal, Canada, Aug. 2001.

[KF10] Koller, D. and Friedman, N., “Probabilistic Graphical Methods: Principles and Techniques,” The MIT Press, 2010.

[LW11] Liu, Q., Wu, T., & Luo, X. (2011). A space-time network model based on improved genetic algorithm for airport taxiing scheduling problems. *Procedia Engineering*, *15*, 1082-1087

[M01] Murphy, K., “The Bayes Net Toolbox for MATLAB,” Computing Science and Statistics, Vol. 33, 2001.

[MB09] Murdoch, J., Barmore, B., Baxley, B., Abbott, T., “Evaluation of an Airborne Spacing Concept to Support Continuous Descent Arrival Operations,” Eighth USA/Europe Air Traffic Management Research and Development Seminar.

[N10] National Science and Technology Council, “National Aeronautics Research and Development Plan,” February 2010.

[NE90] Nedell, W., and Erzberger, H., "The Traffic Management Advisor," 1990 American Control Conference, San Diego, CA, May 1990.

[SA14] Stroiney, S., et. al., “SOSS Input File Modifications to Support Terminal Airspace Modeling,” Project Internal Design Document.

[SC11] Schleicher, D., Clarke, J-P., Saraf, A., et. al., “A Concept of Operations for a NextGen Metroplex Scheduling Concept,” AIAA Aviation Technology, Integration, and Operations Conference, AIAA 2011-7065, Virginia Beach, VA, 2011.

[SL13] Stroiney S., Levy B., Khadikar H., Balakrishnan H., “Assessing the Impacts of JFK Ground Management Program,” Digital Avionics Systems Conference, Syracuse, NY, 2013

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

[W12] Windhorst, R. (2012). Towards a Fast-time Simulation Analysis of Benefits of the Spot and Runway Departure Advisor. In *14th Aviation Technology, Integration, and Operation (ATIO) Conference*.

Appendix A: SOSS—Simulation Platform for Proof-of-concept Simulation Experiments

As mentioned previously, we used simulations of airport surface and terminal airspace traffic in order to provide a proof-of-concept for PROCAST. We used a NASA airport surface traffic simulation platform called Surface Operations Simulator and Scheduler (SOSS) for this purpose. In order to enable seamless simulation of aircraft movement from gate to metroplex terminal airspace boundary, we made some enhancements to the SOSS simulation platform. This appendix provides a brief background on SOSS and its capabilities, and then describes the capability-enhancements that we made to SOSS.

A. Background

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 [W12].

B. SOSS Enhancements—Design of Terminal Procedures

We derived terminal procedures for the JFK terminal airspace from recorded surveillance data. The routes taken by arrivals from each arrival fix to each runway, and the routes taken by departures from each runway to each departure fix, were determined by examining surveillance tracks for a sample of flights.

We used surveillance data recorded by the Saab Sensis Aerobahn system operating at JFK airport. This system records position data for flights operating on the airport surface and in the terminal area. Our data sample was taken over a period of several months from Fall of 2013.

We first restricted the data sample to time periods when JFK was operating in the runway configuration we simulated in SOSS – arrivals on 31R and 31L and departures on 31L. For each flight in this sample, we determined its arrival or departure runway, as well as the arrival or departure fix which it passed most closely. We separated flights into one sample for each fix-runway or runway-fix pair. The set of arrival and departure fixes was determined by those already defined in SOSS, as well as a few other fixes commonly observed in ASDI data.

For each fix-runway or runway-fix pair, route clusters within the surveillance data were identified and less common flight paths were filtered out. We used an in-house tool for identifying the terminal route clusters. The clusters were compared to arrival and departure procedures published for JFK. The procedure that most closely matched the route cluster was used as the basis for the eventual Terminal Procedure route (see Figure 12). Fixes that comprised the Terminal Procedure routes included those called out in the published procedure and intermediate fixes chosen to add definition to the route where needed. In the end, this process produced a route as a sequence of latitude-longitude waypoints extending from arrival fix to runway or from runway to departure fix.

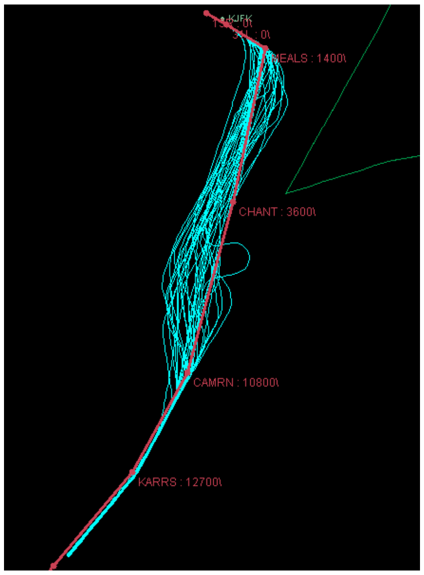


Figure 12. Terminal airspace route identification using trajectory clustering and mapping to a standard terminal procedure.

C. Modifications to SOSS Input Files

To enable SOSS to simulate the terminal area, several modifications were required in the SOSS input files. Two categories of changes were required. First, airport adaptation files must contain information about the airspace, defining routes from each departure runway to each departure fix and from each arrival fix to each arrival runway. Second, scenario files must contain, for each flight, information about the route to take in the terminal airspace.

To encode the routes in the terminal airspace, we first added all waypoints found on these routes to the file fixes.txt. In addition to the already-present arrival and departure fixes, we added a third type, “LATLON_FIX”. Every waypoint identified on a route in the previous step had a corresponding entry in this file. Next, we created a new file called air_routes.txt. Just as routes.txt describes routes on the surface, air_routes.txt contains a list of waypoints (referencing fixes.txt) for each named route. We defined one route for every arrival-fix-runway and runway-departure-fix pair.

We also modified the scenario definition file (.list_data) to tell SOSS which terminal area route to use for each flight. An extra field was added for every flight, containing the name of the route, referencing air_routes.txt. Also, for arrivals the START TIME (i.e. P-time) field was redefined as the time crossing the arrival fix (rather than the time starting final approach to the runway).

These input file modifications are described in more detail in [SA14].

When creating a SOSS scenario, we used a few sample days from the Aerobahn dataset used previously to define the terminal area procedures. We determined the runway and fix for each flight from the surveillance data.

D. Modeling of Aircraft Ascent and Descent in the Terminal Airspace

With airborne routes defined, the next step involved modifying SOSS to simulate aircraft movement in the terminal airspace. A kinematic model similar to that already used for aircraft propagation on the airport surface was used to model aircraft transit from one airborne fix to the next. In order to create an initial schedule at each fix, kinematic equations were also used to determine the time to traverse the distance between fixes. The implemention of these features involved enhancing functionality in existing states in the SOSS Aircraft Class state machine to include the necessary classical mechanics calculations. As arrivals enter the scenario, the aircraft state AC_VECTOR_FOR_DESCENT was modified. For departures leaving the runway, the aircraft state AC_ASCENT was modified. Additional functions were added and called during aircraft initialization in order to calculate nominal schedule times for the fixes along the airborne route. These changes to SOSS enabled the simulation of aircraft movement along routes defined within the terminal airspace.

E. Modifications to the Interface Between SOSS and External Schedulers

With terminal area procedures being modeled, it was necessary to modify the interface between SOSS and external schedulers (the Common Algorithm Interface, or CAI).

The first change was to re-interpret the "activation time" for arrivals as the time crossing the arrival fix, rather than the time starting final approach to the runway – just as we did in the scenario definition file.

The second change was to add all airspace waypoints to the interface, just as surface nodes were included. In particular, the route for a given flight includes airspace waypoints before surface nodes for arrivals, or after surface nodes for departures. The "next node" for a given flight will be an airspace waypoint if the flight is airborne. An airspace node is distinguished from a surface node by adding an "F" before its index. For instance, an arrival route might be "AIR F23 F2 F17 F1 F35 822 1002 23 17 32 GATE" where "F35" is the last airspace waypoint and "822" is the first surface node.

For data returned by the scheduler to SOSS, we again treat airspace fixes just like surface nodes. The scheduler can set an STR for any airspace fix, just like for a surface node.

F. Modifications to SOSS Output Files

We modified the SOSS text output files to include kinematic and scheduling information for the airborne portion of the trajectory.

ACStateHistoryData.txt includes kinematic state for both the airborne and surface portions. We added status codes "AC_VECTOR_FOR_DESCENT" and "AC_ASCENT" to be used while the aircraft is airborne.

ACScheduleData.txt treats airspace waypoints just like surface nodes. As in the CAI, the index for a surface node is preceded by an "F". For each waypoint and node, the file provides the arrival time at the waypoint/node, the departure time, and the scheduled time of release.

Appendix B: Agile Integration Process

Integration of different sub-components of PROCAST (i.e., individual software modules such as NACRE, BBNs and SOSS) produced by different sub-project teams into a single, unified system was a challenging task. There were three different development efforts progressing in parallel during the middle-phase of the project—(i) developing a MATLAB-based simulation framework that included BBN-based prediction capability, (ii) enhancing NASA's SOSS simulation platform for extending its simulation capabilities to cover the terminal airspace, and (iii) developing the NACRE genetic algorithm for optimizing 4D trajectories. These three development processes were interdependent and were being developed by geographically distributed teams. The modeling requirements for any two processes usually changed based on the problems faced in the development of the third process and modeling decisions made for resolving those problems. Hence, the team adopted an Agile integration process for ensuring timely, correct, and seamless integration of these sub-components, which were being developed in parallel by geographically distributed sub-teams, as well as for managing changing modeling requirements.

Figure 13 shows a flowchart of the Agile integration process. The team worked on an N-day sprint. We started the work with N=14 days (i.e., a two-week cycle). But, we increased the frequency of collaboration later on by reducing the cycle time to 7 days. On the morning of the first day in the cycle, the entire team met for half an hour to discuss enhancement plans for the upcoming cycle. Each of the three main development tasks—NACRE, SOSS, and MATLAB framework—were assigned a lead who was responsible for implementing all the agreed-upon changes for that cycle. MATLAB framework development was a complicated task, and hence it was broken down into manageable parts—BBNs, NACRE Sim Executor, and Book-keeping. Individuals were assigned to each of the idenitified high-priority tasks. Finally, at noon time on the last working day of the cycle, everybody released their latest version of the software they were working on. The individual pieces of code were versioned and integrated into a single Subversion code repository. The team then performed a series of quick unit tests on the newly developed modules, followed by integration testing. Testing was completed in half a day. At the end of testing, we were able to identify unfinished requirements, new requirements and sometimes remove earlier requirements that have been deemed unnecessary by changes in modeling approaches. On the next day, we started the next cycle by again meeting and discussing progress from previous cycle (unfinished requirements, new requirements, new tasks, etc.) and enhancement plans for this cycle.

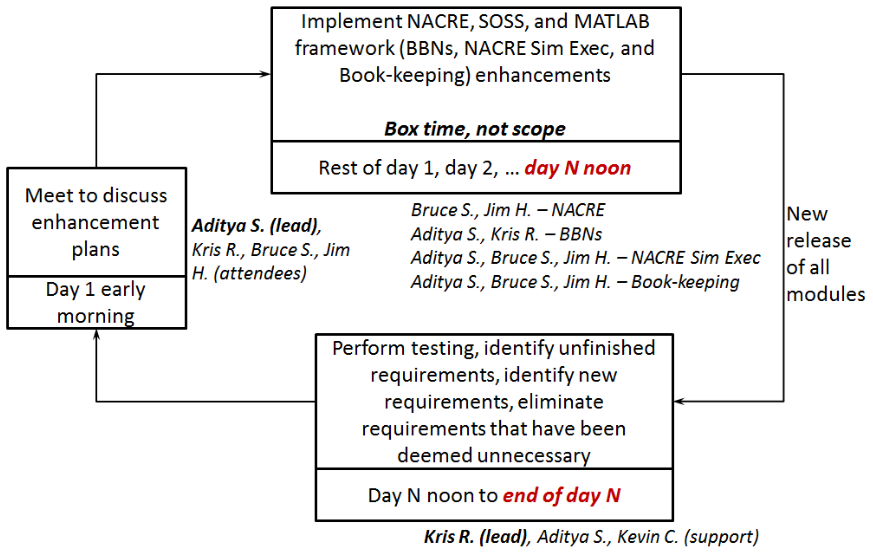

Implement NACRE, SOSS, and MATLAB framework (BBNs, NACRE Sim Exec, and Book-keeping) enhancements
Box time, not scope
Rest of day 1, day 2, … day N noon
Bruce S., Jim H. – NACRE
Aditya S., Kris R. – BBNs
Aditya S., Bruce S., Jim H. – NACRE Sim Exec
Aditya S., Bruce S., Jim H. – Book-keeping
New release of all modules
Meet to discuss enhancement plans
Day 1 early morning
Aditya S. (lead), Kris R., Bruce S., Jim H. (attendees)
Perform testing, identify unfinished requirements, identify new requirements, eliminate requirements that have been deemed unnecessary
Day N noon to end of day N
Kris R. (lead), Aditya S., Kevin C. (support)

Figure 13. Agile management process applied to the integration of PROCAST components.

The Agile software integration process provided us with useful and timely feedback on correctness and integration issues of the different components throughout development, and helped reduce the development time.