SOFTWARE TOOLS TO SUPPORT RESEARCH ON AIRPORT DEPARTURE PLANNING

Francis Carr, Antony Evans, Eric Feron, John-Paul Clarke

International Center for Air Transportation

Massachusetts Institute of Technology

Cambridge MA

Abstract

A simple, portable and useful collection of software tools has been developed for the analysis of airport surface traffic. The tools are based on a flexible and robust traffic-flow model, and include calibration, validation and simulation functionality for this model. Several different interfaces have been developed to help promote usage of these tools, including a portable Matlab™ implementation of the basic algorithms; a web-based interface which provides online access to automated analyses of airport traffic based on a database of real-world operations data which covers over 250 U.S. airports over a 5-year period; and an interactive simulation-based tool currently in use as part of a college-level educational module. More advanced applications for airport departure traffic include taxi-time prediction and evaluation of "windowing" congestion control.

Introduction

Congestion and delays on the airport surface are a significant source of financial and uncertainty costs for both airlines and the flying public. These mounting costs have fueled research into methods for improving the airport departure process, either via the elimination of existing inefficiencies or the creation of decision-aiding tools. Lacking any means to field-test new hypothesized models and proposed improvements, this research has necessitated the use of software tools for modeling, analyzing and simulating airport surface traffic.

A variety of software tools with generally similar goals are commercially available and widely used for airport planning [1]. However, as often the case, commercial software tools lack several characteristics which are required in academic research:

- Easy to modify the traffic-flow model used by the software.
- Traffic-flow model and software are both well-adapted to limitations of *available* real-world operations data.
- Software is low-cost, both in terms of licensing fees and the manpower needed to use it, with a fast learning curve.

These required characteristics have led to the development of several in-house software tools. This paper gives an overview of these software tools, including their basic functionality and the underlying algorithms.

In addition, two interfaces to the software tools have been developed to help promote usage of these tools. One interface is web-based and allows online users to access a large database of real-world operations data and then download airport traffic analyses generated by our server. Through continuing improvements to this interface, new applications such as taxi-time predictors and proposed congestion-control algorithms are being made available in a useful public forum, allowing these research results to move more easily out of the laboratory and into the hands of users in the field. The second interface is part of an interactive educational module currently in use at MIT. This interface consists of a set of Excel™ spreadsheets which contain airport operations data; students can modify this data and then feed their proposed schedules and traffic restrictions into a software tool which simulates the expected airport surface traffic resulting from their changes.

Software Tools

Description Of Input And Output

The traffic-flow model underlying the software treats the airport surface as a "black-box" queueing system with observed inputs and outputs, and a simple set of internal dynamics which are not observed directly but are assumed to have a known structure[1]. Hence, to be useful, real-world operations data must provide a description of airport surface traffic which appeals to the model. For each aircraft, the following information is required:

1) ***source***: a "classifier" variable, describing the origin of the aircraft on the airport surface. It is assumed that different sources of aircraft (e.g. airlines, terminals, or gates, depending on the level of detail provided by the real-world operations data) will have different characteristic taxi-times. All of the software tools account for this effect.
2) T_i (***T***ime-***i***n): epoch when the aircraft enters the system (i.e. pushback for departure traffic, or wheels-on for arrival traffic).
3) T_o (***T***ime-***o***ut): epoch when the aircraft exits the system (i.e. wheels-off for departure traffic, or gate-arrival for arrival traffic).

A variety of real-world operations data can be used to obtain these measurements. Airline operations centers often maintain records of gate-arrival and pushback times. Multilateration and terminal-area radar systems can estimate wheels-off and wheels-on times. In our research, we have used operations data taken from either the Airline Service Quality Performance (ASQP) database or directly from FAA tower flight-strips.

The description of airport surface traffic provided by ASQP is relatively coarse. The source of each aircraft is only partially approximated by the airline. Taxi-paths, runway assignments, and gate assignments are not recorded. A fraction of the total airport surface traffic goes unmonitored, since ASQP only records domestic jet operations by the major U.S. passenger carriers, ignoring all international flights, all prop and regional-jet traffic, and all traffic by regional airlines and freight carriers. However, the ASQP data does have several advantages: ASQP records the out-off-on-in (OOOI) epochs with reasonable 1min accuracy and precision, and is available for hundreds of U.S. airports over a period of several years.

For applications requiring finer detail than ASQP provides, good results have been obtained using data from FAA tower flight-strips [2]. When combined with tower and TRACON logs, flight-strip data provides a very thorough and complete description of traffic at a given airport. There can be some difficulty obtaining flight-strip data, although for research purposes it is perfectly reasonable to use flight-strip data which is several months old. Flight-strip data can also be time-consuming to transcribe from paper to the computer, although ongoing efforts to increase automation in the tower and TRACON may mitigate this problem.

Once input data has been obtained, it is fed to the software tools, which produce several types of outputs. The calibration algorithms produce a set of descriptive statistics for the airport surface traffic described by the real-world operations data. Such statistics are then used together with software tools for Monte Carlo simulation to estimate statistics for taxi-times, congestion and delays. Customized visualization tools are available for most of these results.

Algorithms

The software tools are based on a small set of basic algorithms. The calibration algorithms take raw input data and produce calibrated model parameters; these parameters include the estimated mean, variance and distribution of unimpeded taxi-time, and the maximum rate at which the traffic bottleneck can service aircraft. The simulation algorithms take a calibrated model together with a set of system inputs (e.g. pushbacks in the case of departure traffic) and estimate the expected system outputs (e.g. wheels-off times). Finally, the validation algorithms combine calibration and simulation to determine whether a calibrated model actually reproduces the traffic behaviors observed at a particular real-world airport.

[1] See [6, 7] for information on the structure of the traffic-flow model.

Calibration Of Unimpeded Taxi-Time

Let $\boldsymbol{S}$ denote the set of sources of aircraft, as specified in the input data. One quantity of interest is the traffic statistics for aircraft originating at a given source $s \in S$, for example the mean and standard deviation of taxi-time for departures from a given airport terminal. While the mean and standard deviation are useful statistics, they do not completely characterize the variability of taxi-times. Instead, the following algorithm is used to directly estimate probability density functions (p.d.f.'s) of unimpeded taxi-time:

1) From the input data, reconstruct a time-history of the number of aircraft on the airport surface.
2) For each source $\boldsymbol{s}$, consider the set of aircraft originating from that source. Filter these aircraft, selecting a subset which entered the system under conditions which imply little to no interaction with congestion at the bottleneck queue.
3) Using standard statistical techniques, estimate a p.d.f. for unimpeded taxi-out time using this subset, under the assumption that each aircraft in the subset represents an independent and identically distributed sample from the p.d.f.

This basic formulation is used by most software tools currently used to predict or estimate taxi-times. Different software tools use different methods in assigning sources to aircraft, and in selecting the subset which interacts least with the bottleneck queue. For example, the basic algorithm used in the Surface Movement Advisor (SMA) tool at ATL to predict departure taxi-times assumes a single source (all aircraft are treated identically); a subset consisting of the last twelve aircraft to push[2]; and a simple interaction filter which removes the highest and lowest taxi-times from each 12-aircraft subset [3]. Another algorithm is described in the documentation for the Consolidated Operations and Delay Analysis System (CODAS) [4]. CODAS estimates unimpeded taxi-times by assigning sources according to season of the year and airline carrier; computing a linear regression between taxi-time and queue-lengths at time of pushback using both arrival and departure queues; and selecting a "virtual" subset by extrapolating the linear regression to zero queue-lengths.

Our software tools leave the method of assigning sources entirely to the input data: the classification variables specified in the input records are used in the calculations. The web-based interface (described below) uses airlines as sources, since airlines typically cluster their gates, and because no better indication for an aircraft's source is available in ASQP. The Excel™-based interface (also described below) comes pre-packaged with a set of EWR operations data, and maps different airlines and flight-numbers to the 4 airport terminals at EWR.

An accurate heuristic for determining the interaction between an aircraft and the congested queue near the airport's primary bottleneck has been developed in [5,6,7]. For each aircraft, a traffic-interaction index[3] is computed. This index counts the number of system-exits (takeoffs for departure traffic, or gate-arrivals for arrival traffic) which occur while that aircraft is on the airport surface. For a given aircraft $\boldsymbol{F}$, a low index indicates that $\boldsymbol{F}$ could not have waited in a long queue, else a large number of other aircraft would have left the system while $\boldsymbol{F}$ was taxiing. Similarly, $\boldsymbol{F}$ could not have experienced long delays unrelated to traffic, else a large number of undelayed aircraft would have passed $\boldsymbol{F}$ on the tarmac and left the system while $\boldsymbol{F}$ was taxiing. This index is computed for each aircraft in the input dataset, and a subset of aircraft with "small" indices is used to estimate the taxi-time p.d.f.'s. The web-based interface uses a threshold of _ the median index to determine which aircraft have "small" indices.

Calibration Of Bottleneck Service Rate

Methods to estimate the service-rate of the runway have been the subject of intensive investigation [5,8,9,10]. The central problem of these analyses is to observe or compute the maximum throughput of the runway system. Many of these analyses are based on the same idea: estimate the probability that the runway queue is

[2] Note that this lumps together unimpeded taxi-time with queueing effects, a significant difference from the traffic-flow model assumed in this paper.

[3] This index of an aircraft's interaction with other traffic is denoted N_H or Q in the given references.

empty, and then focus on intervals when that probability is low. Note that this idea generalizes to traffic bottlenecks other than the runway. A number of airport improvement programs are underway to automate real-time surveillance of surface movements, typically focusing on multilateration systems. As improved operations data from these surveillance systems become available, it will become useful to consider additional surface bottlenecks such as crossing-points and merges along the taxiways. Hence the calibration algorithm has a flexible design which only assumes a simple FIFO structure for the bottleneck queue, rather than the specific structure observed at the runway queue.

The calibration algorithm is based directly on the idea of estimating the probability that the bottleneck queue is empty. Let P_t be the probability that the bottleneck queue is empty at time t. Assume that an estimate has already been obtained for the unimpeded taxi-time p.d.f.'s. for aircraft originating from each source. Initially, before any aircraft are considered, the queue is presumed to be empty ($P_t \equiv 1$). For each aircraft F, for all $t \in [T_i, T_o]$, perform the following update:

$$P_t := P_t \times \mathbf{Pr}(\text{taxi-time} \geq t - T_i)$$

This update relies on the assumption that unimpeded taxi-times are independent to justify the multiplication. The probability **Pr** is calculated as the *complementary* distribution function of unimpeded taxi-time, except for $t = To$ when the probability is set to zero, since the queue cannot be empty immediately preceding an aircraft's exit from the system. After iterating through the set of all aircraft, P_t contains an estimate of the probability that the queue is empty at time t. This estimate then yields a set of data consisting of pairs (P_t, N_t^O) where N_t^O is the number of aircraft exiting the system at time t. The statistic of interest is the distribution of N_t^O as $P_t \to 0$, which can be estimated via standard regression techniques.

Simulation

Once the calibration algorithms have been run on a representative sample of input data, the resulting calibrated model dynamics form a well-specified stochastic system which is amenable to Monte Carlo simulation. Two types of Monte Carlo simulation are useful. The calibrated model dynamics can be simulated "as-is" without modification. Actual flight records are used to provide realistic inputs to the system, and the Monte Carlo simulation approximates the expected system outputs. This type of simulation is useful to determine whether the assumed model dynamics in fact reproduce (in an aggregate statistical sense) the actual system dynamics. Another type of simulation involves modifying the model dynamics and then simulating the modified system. This type of simulation is useful to test the effect of incorporating more complex and detailed dynamics[4] or to test the imposition of control schemes designed to optimize some performance metric of surface traffic[5].

Validation

The validation technique is quite simple:

1) Compute a calibrated model using input data from odd-numbered weeks.
2) Using the calibrated model dynamics, driven by input data from the even-numbered weeks, compute several Monte Carlo simulations.
3) Compute a calibrated model using the simulated flight-data produced by the Monte Carlo simulations.
4) Compare the two calibrations.

Simulation of the unmodified dynamics can be vectorized for efficient implementation in Matlab™. For each source in S, a vector of unimpeded taxi-times is generated using standard Monte Carlo techniques from the calibrated p.d.f.'s. Similarly, a vector of bottleneck queueing service-times is generated. For each aircraft, the unimpeded taxi-time is added to the T_i epoch to obtain a simulated qE (**q**ueue-**E**ntry) epoch. Aircraft from all sources are then sorted according to the qE epochs. For a set containing N aircraft, assume the qE epochs are ordered such that:

$$qE(1) \leq qE(2) \leq \ldots \leq qE(N)$$

Then only a single pass through the input data is required to simulate the standard FIFO queueing dynamics [11]:

[4] See [2] for a study on the effect of downstream restrictions.
[5] See [6,14,15] and the section below on "windowing" congestion control for studies of the effect of using gate-holds to mitigate runway queueing.

$T_o(0) \equiv -\infty$

$T_o(n) = \max\{qE(n), T_o(n-1)\} + service\text{-}time(n)$

After simulating the T_o epochs, the aircraft are returned to their original order. A straightforward coding of this algorithm in Matlab™ requires 40 lines of code and can simulate a month's worth of traffic data from a major U.S. hub in <1s on an average desktop computer.

Matlab ™-Based Interface: Basic Tools

The basic software tools are coded in Matlab™ using standard toolboxes. Real-world operations data is input as a single matrix, with columns containing (***source***, T_i, T_o) data, and a row for each aircraft. A well-documented set of functions operate on this singe input matrix to perform tasks including data preprocessing; estimation of model parameters; simulation of the basic model dynamics; validation; simulation-based optimization; and visualization. Matlab™ was chosen as the basic language for implementing the software tools because it is widely available and portable, and it hides irrelevant details of programming to focus on the actual algorithms. Thus the current implementation allows the software tools to be easily adopted and improved by users outside our research group, and encourages rapid prototyping and visualization of new ideas and results.

Web-Based Interface: Calibration Tools

A subset of the Matlab™-based tools are accessible through a web-based interface, available at http://icat-server.mit.edu/~fcarr/online_model/. The web-based interface has access to a database of ASQP records covering more than 250 U.S. airports over the period Jan-1995 through Dec-2001. Users are prompted to select an airport and a month-long period of interest. This input starts a Matlab™ computational engine located on our server. The Matlab™ engine pipes in relevant flight-records from the database; runs the calibration functions; and creates both visual and text-based versions of the results (see Figures 1 through 3). In addition, a taxi-time predictor (described in the "Applications" section) is also provided. Planned upgrades to the web-based interface include automated model validation, improved taxi-time predictors, and an evaluation tool for testing a proposed congestion-control algorithm for departure traffic.

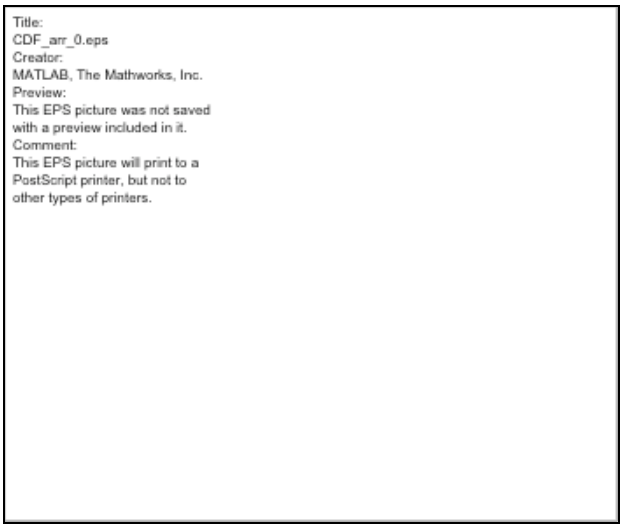

Figure 1: Sample of Visual Output From Web-Based Interface (Calibrated Taxi-Time P.D.F)

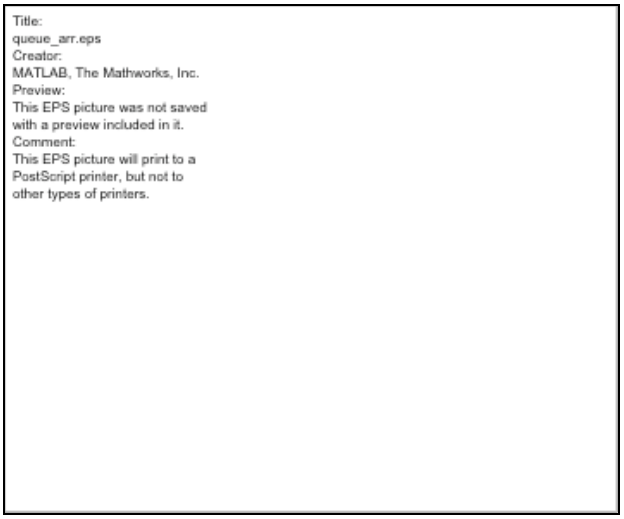

Figure 2: Sample of Visual Output From Web-Based Interface (Bottleneck Rate As $P_t \rightarrow 0$)

```
Arrival statistics:
Fitted parameters for airline 0:
  Mean=6.2727
  Std. deviation=3.0238
  LogNormal \mu=1.3374
  LogNormal \sigma=0.60283
  LogNormal shift=1.7048
Fitted parameters for airline 3:
  Mean=6.7091
```

Figure 3: Sample of Textual Output From Web-Based Interface

Between the time they select an airport and period of interest, and the time they receive their results over the web, online users see a delay of approximately one minute. Once a user has requested calibration data for a particular airport and time-period, the results are archived and immediately available for later downloads, thus removing the one-minute delay.

In practice, the capability to rapidly and automatically baseline the traffic at almost any U.S. airport has proven useful several times in our research. The description of surface traffic as provided by ASQP is relatively coarse, but does adequately describe many of the hub and spoke airports which are of greatest concern to the major U.S. passenger carriers.

Excel-Based Interface: Simulation-Based Optimization

Another subset of the Matlab™-based tools has been packaged as part of an educational teaching module through the Sloan Foundation Systems Studies at MIT [12]. The educational goals of the module are to give students insight into issues which affect airline and air traffic control operations at highly congested airports. To support these goals, the Matlab™-based simulation functions have been re-implemented in Python, a portable, widely-used and cost-free scripting language [13]. Users are provided with a detailed set of Excel™ spreadsheets describing one day of weather and traffic at EWR. Relevant papers and study problems then encourage users to test the effects of different airline schedules and FAA traffic restrictions on departure congestion and delays. Using Excel™, the base EWR data are modified to create different schedules and traffic restrictions. These modified operations data are fed directly to the simulation scripts, which output statistics for each aircraft's taxi-time and queueing delay, as well as statistics on surface congestion for the airport as a whole. These results can then be analyzed and visualized directly in Excel™. A single CD contains the Python installer; the base EWR data; and the simulation scripts.

Currently, parts of this teaching module are used in several classes at MIT, which cover topics including air traffic control, airline operations, and airport modeling and planning. The teaching module is currently under evaluation for similar classroom use at University of Cambridge.

Applications

Taxi-Time Predictors

One of the outputs provided by the web-based interface is a taxi-time predictor originally described in [14]. The output of the taxi-time predictor is a simple lookup table which maps the airline and departure congestion (measured as the number of departing aircraft on the airport surface at T_i) of a given flight to the expected taxi-out time, including queueing delays. This taxi-time estimator can be easily obtained as a byproduct of the data preprocessing which already occurs when the web-based interface is accessed.

Departure taxi-time predictor #2							
(quadratic fit to observed T/Q data)							
Format:							
One traffic source per row (leftmost column)							
One value of N per column (topmost row)							
Smoothed mean of departure taxi-times (minutes)							
	1	2	3	4	5	6	7
Airline #1	9.4	11.3	12.8	14.8	16.4	18.2	20.3
Airline #2	13.6	14.2	14.9	16.2	17.1	18.3	20.1
Airline #3	11.2	12.4	13.5	15.0	16.2	17.7	19.5
Airline #4	12.4	13.2	14.0	15.2	16.1	17.3	18.9
Airline #5	15.2	15.3	15.6	16.5	17.1	18.0	19.5
Airline #6	16.8	16.3	16.1	16.6	16.7	17.3	18.5

Table 1: Sample Of Output From Web-Based Interface (Taxi-Time Predictor)

Windowing Congestion Control

Windowing congestion control was first proposed in the context of airport surface departure traffic in [15] under the name of "N-control". The basic idea is intuitively appealing. If too many departing aircraft are already queued up on the airport surface waiting to take off, then any additional departures released from the gates will necessarily be delayed. Delays taken at the gate are less expensive in several respects than delays taken while taxiing. Hence, if departure surface congestion is too high, it may be beneficial to temporarily delay additional pushbacks at the gate.

Gate-delay is certainly less expensive than taxi-out-delay in terms of environmental emissions; direct operating costs for crew and fuel; and missed opportunities to catch late passengers and baggage. However, there are also potential disadvantages due to increased utilization of scarce gates; missed opportunities to catch an early position in the runway queue; and degraded on-time departure statistics. The optimal balance among these costs is not immediately apparent, which has led us to develop software tools which enable a tradeoff analysis between the competing factors.

The quantities computed in the tradeoff analysis include total gate delay, total taxi-out delay, and aggregate gate utilization. Two control parameters are used: the maximum number of departing aircraft allowed on the airport surface (denoted $\boldsymbol{C}$), and the maximum gate-delay assigned to any departing aircraft (denoted $\boldsymbol{D}$). It is assumed that the max-delay constraint is "stronger" than the max-congestion constraint, so that departing aircraft are never held longer than $\boldsymbol{D}$ minutes under any circumstances. Other inputs to the tradeoff analysis include the predicted pushback schedule over the period of interest, and a set of calibrated model parameters as produced by the calibration algorithms.

The tradeoff analysis is computed via Monte Carlo simulation. In general, the simulation algorithm is similar to the algorithm previously described for simulation of the unmodified dynamics, including a vectorized computation of taxi-out and runway-service times for each flight, followed by a single pass through the flight records. However, the simulation dynamics must be modified to account for the control constraints.

A priority queue $\boldsymbol{TW}$ is used to track aircraft currently taxiing towards the runway queue. The following set of updates are iterated until all pushbacks recorded in the input data have occurred and $\boldsymbol{TW}$ is empty:

1) While some aircraft have not pushed and fewer than $\boldsymbol{H}$ aircraft are in $\boldsymbol{TW}$, allow an aircraft to push at its desired pushback time $\boldsymbol{T_i}$. This aircraft is placed in $\boldsymbol{TW}$ according to the time it should reach the runway queue (denoted $\boldsymbol{qE}$ in accord with the notation used previously).

2) While some aircraft have not pushed and the next aircraft to push would have to wait too long before a takeoff, allow the next aircraft to push at time $\boldsymbol{T_i}+\boldsymbol{D}$, thus reaching the runway queue at $\boldsymbol{qE}+\boldsymbol{D}$.

3) If some aircraft have not pushed and exactly $\boldsymbol{H}$ aircraft are in $\boldsymbol{TW}$, allow the next aircraft to push at time $\max\{\boldsymbol{T_i}, \text{next } \boldsymbol{T_o}\}$

4) If $\boldsymbol{TW}$ is not empty, remove the earliest aircraft from $\boldsymbol{TW}$. Among the set of aircraft currently taxiing out, this aircraft will be first to reach the runway queue and take off. Its takeoff time can be computed using the standard FIFO queueing dynamics discussed previously.

While complex at first glance, these modified simulation dynamics allow both the max-congestion and max-delay constraints to be imposed on the basic simulation dynamics without losing the simple single-pass computational strategy. A profile of the Matlab™ execution of this algorithm indicates that almost 90% of the computational time is spent modifying the priority queue data structure; recoding this data structure into compiled C code which is called from Matlab™ substantially reduces the execution time to <1s.

The modified simulation dynamics have several desirable characteristics. Aircraft still push back in their original order to preserve fairness among competing airlines. No aircraft is held beyond a given threshold of delay[6], and thus it is

[6] Note that only a trivial modification is required to make the max-delay threshold $\boldsymbol{H}$ specific to each aircraft, thus allowing much greater planning flexibility.

easy to ensure that on-time departure performance is not unduly impacted. Different control levels can be rapidly evaluated to determine the expected performance in terms of congestion and delay, thus allowing airline operation centers and air traffic controllers to make realistic tactical decisions.

The main drawback to the modified simulation dynamics is that gate utilization is not properly accounted for. It is easy to track the *aggregate* gate utilization by tracking the cumulative count of pushbacks and gate-arrival events as they occur, and preliminary studies at EWR and ATL indicate that windowing congestion control does not typically require additional gates beyond those which each airline already uses [16]. However, gates are obviously not interchangeable. To determine whether windowing congestion control will delay a pushback until a later gate-arrival is impacted, detailed operations data including planned and feasible alternate gate assignments for each flight is required.

References

[1] Khan, Kashif et al, November 1997, "Existing and Required Modeling Capabilities for Evaluating ATM Systems and Concepts", MIT International Center for Air Transportation, Cambridge MA, http://web.mit.edu/aeroastro/www/labs/AATT/

[2] Carr, Francis, Antony Evans, J.P. Clarke, Eric Feron, May 2002, "Modeling and Control of Airport Queueing Dynamics under Severe Flow Restrictions", Proceedings of 2002 American Control Conference, IEEE 02CH37301C.

[3] Glass, Brian et al, U.S. patents #6,278,965 "Real-time surface traffic adviser" and #6,161,097 "Automated traffic management system and method", United States Patent and Trademark Office, http://patft.uspto.gov/.

[4] Office of Aviation Policy and Plans, September 1997, *Documentation For The Consolidated Operations And Delay Analysis System*, FAA, Washington, D.C., pp. 25-28.

[5] Idris, Husni, 2000, *Observations and Analysis Of Departure Operations At Boston Logan Airport*, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA.

[6] Carr, Francis, February 2001, *Stochastic Modeling And Control Of Airport Surface Traffic*, Master's thesis, Massachusetts Institute of Technology, Cambridge MA.

[7] Andersson, Kari, Francis Carr, Eric Feron, William Hall, 2000, "Analysis And Modeling Of Ground Operations At Hub Airports", 2000 USA-Europe Air Traffic Management Seminar, Naples Italy.

[8] Peterson, Michael, Dimitris Bertsimas, Amedeo Odoni, August 1995, "Models And Algorithms For Transient Queueing Congestion At Airports", Management Science, 41(8) August 1995, pp. 1279-1295.

[9] Gilbo, Eugene, September 1993, "Airport Capacity: Representation, Estimation, Optimization", IEEE Transactions on Control Systems Technology, 1(3) September 1993, pp. 144-153.

[10] Abundo, Stephanie, June 1990, *An Approach For Estimating Delays At A Busy Airport*, Master's thesis, Massachusetts Institute of Technology, Cambridge MA.

[11] Gallager, Robert, 1998, *Discrete Stochastic Processes*, Kluwer Academic Publishers, Boston MA, pp. 202.

[12] Evans, Antony, J.P. Clarke, June 2002, "Responses to Airport Delays – A System Study of Newark International Airport", Report No. ICAT-2002-5, MIT International Center for Air Transportation, Cambridge MA.

[12] "Python homepage", http://www.python.org.

[13] Idris, Husni, J.P. Clarke, Rhani Bhuva, Laura Kang, 2002, "Queueing Model for Taxi-Out Time Estimation", Air Traffic Control Quarterly, 10(1) 2002, pp. 1-22.

[14] Pujet, Nicolas, Bertrand Delcaire, Eric Feron, 2000, "Input-Output Modeling And Control Of The Departure Process Of Busy Airports", Air Traffic Control Quarterly, 8(1) 2000, pp. 1-32.

[15] Carr, Francis, October 2000, unpublished research work, http://icat-server.mit.edu/~fcarr/…/Publications/N-controlForEurocontrol.pdf.