Optimal Control of Airport Operations with Gate Capacity Constraints

Harshad Khadilkar and Hamsa Balakrishnan

***Abstract*— The mitigation of airport surface congestion is an important step towards increasing the efficiency of the air transportation system, and decreasing flight delays. This paper proposes a strategy to control the release of departing flights from their gates with the specific objective of reducing their taxi times and fuel consumption, while limiting the impact on airport throughput. The proposed strategy also explicitly accounts for the practical constraints that arise due to limited gate resources at the airport. A stochastic network abstraction of the airport surface is used to model aircraft movement, and the optimal release time for each aircraft is calculated using dynamic programming. Simulations of operations at Boston's Logan International Airport in the US are used to illustrate the advantages of the proposed policies.**

I. Introduction

Airport surface congestion is one of the major problems faced by the air traffic system, and results in a significant amount of aircraft fuel consumption and emissions even before takeoff. Since the total amount of surface fuel burn is roughly proportional to the taxi times of aircraft [1], reducing aircraft taxi times significantly reduces fuel consumption. A promising congestion mitigation approach is to hold aircraft at their gates until it is optimal for them to start taxiing, as was demonstrated in [2]; however, limited gate availability can pose a challenge to the implementation of such protocols at major airports. It is therefore important to account for such constraints when designing congestion control strategies.

Several studies have found that holding aircraft at the gate when an airport is experiencing congestion can help reduce taxi times and fuel burn. The proposed protocols range from pure gate-holding [3], [4], [5], [6] to explicit control of surface movement [7], [8], [9]. In contrast to aircraft in the departure queue at a runway, aircraft waiting at the gate have their engines turned off. These aircraft encounter lower congestion on the surface, thus reducing their taxi times. The primary aim of these prior studies was to limit surface congestion, and to then evaluate the incidental benefits in fuel burn [2]. Constraints such as the availability of gates at the airport have not been explicitly considered in literature. However, these factors are important in practice, especially when departures are being held at the gate. Arriving aircraft that are waiting for an occupied gate to be vacated can block active taxiways and/or alleyways,

Harshad Khadilkar is a doctoral candidate in the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA harshadk@mit.edu.

Hamsa Balakrishnan is an Associate Professor in the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA hamsa@mit.edu.

This work was supported in part by NSF through a CAREER Award (ECCS-0745237) and CPS:Large:ActionWebs (Award number 0931843).

Fig. 1. Network model of Boston Logan airport.

which is undesirable at space-constrained airports. This paper focuses on the development of a pushback control strategy that explicitly targets fuel savings and taxi time reduction. At the same time, it balances airport performance with gate availability constraints.

This paper adopts a network abstraction of airport operations, developed in prior work [10]. A *gate* is a parking bay where passengers board and disembark from aircraft. A collection of proximal gates is called a terminal. The network model used in this paper is comprised of gates (*sources*), major taxiways (*links*), and runways (*sinks*). The network for Boston Logan International Airport (BOS) is shown in Fig. 1. Gates at each of the four main terminals are located at nodes 1, 2, 3 and 8 respectively. The runways are at nodes 6, 7, 10, 11, 12, 13, 14 and 15. There are some nodes (for example, 4 and 5), that are neither sources nor sinks, but are intersections of major taxiways.

Pushback is the process of pushing an aircraft back from the gate, in preparation for taxi to the runway. Aircraft do not start their engines until pushback is completed, and therefore do not consume any fuel while at the gate. During this time, electrical power for systems such as air conditioning is derived from auxiliary ground-based sources which consume much less fuel than an idling aircraft engine. *Pushback delay* is an instruction given to an aircraft by the air traffic controller, asking it to delay the start of its pushback process. The *pushback buffer* is the set of aircraft that are currently parked at a given terminal.

Sec. II of this paper describes a stochastic airport taxi model that was developed from surface surveillance data. Sec. III presents the development of a control strategy using

a reduced network model. Sec. IV extends the formulation to the complete airport network, using BOS as an example. The proposed control strategy is shown through simulations to reduce taxi times and fuel burn, while maintaining a balance between surface congestion and gate availability.

II. Model of Aircraft Taxi Times

A set of random processes is used to model the taxi operations of aircraft. The taxi-out time of each aircraft, on each link in the network, is the sum of two random variables: (i) unimpeded taxi time, and (ii) stationary time. The expected taxi-out time over each link increases with congestion, that is, the number of departing aircraft already on the surface when the current aircraft leaves its gate. In the rest of this paper, this number is referred to as the *surface traffic level*, k. For each link l, the expectation of taxi time t_l for a given surface traffic level is

$$\mathbb{E}[t_l|k] = \eta_l + k\,\frac{X_l}{\mu_l}. \tag{1}$$

Here, η_l is a constant denoting the expected taxi time across link l when $k = 0$, that is, the expected unimpeded time. The term $\frac{X_l}{\mu_l}$ is also constant. The expected time of each individual stop on the link is given by $\frac{1}{\mu_l}$, while X_l defines the sensitivity of the number of stops to the surface traffic level. The total expected taxi-out time for a given aircraft is calculated by summing the expected taxi times on all the links in its path. The taxi path is assigned by the air traffic controller, and is assumed to be known beforehand.

III. Single-Link Control Strategy

This section develops the control strategy for a simplified airport model, where it is assumed that the network is composed of only one link. A set of gates (the pushback buffer) is located at the source node of the link, and the runway is located at the sink node.

A. Simplified model description

Consider the single-link network with taxi time parameters η, X and μ as described in Sec. II. If this link is in steady state at a traffic level of k, the average taxi time across it is given by Eq. (1), and the average inter-departure time from the link is

$$\Delta t_k = \frac{\mathbb{E}[t_l|k]}{k} = \frac{\eta}{k} + \frac{X}{\mu}. \tag{2}$$

The minimum inter-departure time is achieved as $k \to \infty$, and is given by $\Delta t_\infty = \frac{X}{\mu}$. This value characterizes the theoretical maximum throughput of the link, but corresponds to an infinite expected taxi-out time. The model predicts that this maximum throughput will be achieved asymptotically. This performance saturation is in agreement with empirical studies [2], [11], which are based on operational airport data.

The single-link case assumes that there is a single terminal with a corresponding pushback buffer that holds all the aircraft at the airport. The setup is illustrated in Fig. 2. Aircraft enter the buffer once they land at the airport and pull into their gates. This arrival process is assumed to be

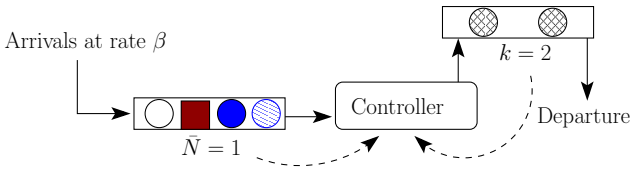

Fig. 2. Illustration of the single-link model with a single buffer. The hashed circle denotes the aircraft that is scheduled to push back next, the solid circle denotes a gate that is occupied-active, and the solid square denotes a gate that is occupied-inactive. Aircraft that are actively taxiing are denoted by double-hashed circles.

Poisson with a rate β. Note that the gate arrival process is stochastic even at real airports. Uncertainty in gate arrival times is introduced both by errors in predicted landing times as well as by the variability in taxi-in times. A probabilistic rate is a robust way to channel arrival information to the departure control algorithm. On arrival to the gate, each aircraft begins a turnaround process with loading and unloading of passengers and cargo. Gates containing aircraft that are being turned around are tagged as being *occupied-inactive*. Once this is completed, these aircraft call the air traffic controller for permission to pushback and are tagged as being *occupied-active*. The gate capacity of the terminal is denoted by $N_{\max}$. The available gate capacity (denoted $\bar{N}$) is therefore the total number of gates ($N_{\max}$) less the number of gates currently occupied, and takes values $\bar{N} \in \{0, 1, \ldots, N_{\max}\}$. If an aircraft arrives when all gates are occupied, it is accommodated by the immediate release of a gate-held aircraft from the pushback buffer.

B. System dynamics

As described in Sec. III-A, the surface traffic level k drives the taxi-out times, while the available gate capacity $\bar{N}$ governs the maximum alowable gate delays. The state of the system is defined by the pair $(\bar{N}, k)$. Control is implemented by assigning a delay of u time units to the first aircraft in the pushback buffer, thereby maintaining a First-Come-First-Served (FCFS) order. Two independent stochastic processes run during this time interval u: (i) Aircraft arrive as a Poisson process with rate β, and (ii) departures that have been previously released depart from the link. If the epochs are defined by the instant of each pushback, state transitions between successive epochs are stochastic. Pushback delay (the control input) is assigned to the next aircraft in the pushback buffer at the beginning of each epoch.

Let $p_{\theta_1\theta_2}(u)$ denote the transition probability from state $\theta_1 = (\bar{N}_1, k_1)$ to $\theta_2 = (\bar{N}_2, k_2)$ after a time u. Since $\bar{N}$ and k are governed by two independent random processes, the state transition probability can be decomposed into the probability of transition from $\bar{N}_1$ to $\bar{N}_2$ and the one from k_1 to k_2 as

$$p_{\theta_1\theta_2}(u) = p_{\bar{N}_1\bar{N}_2}(u)\, p_{k_1k_2}(u). \tag{3}$$

The first term in the right hand side is easy to calculate, since it is governed by a Poisson process of rate β. The second term is more difficult to estimate, since aircraft already on the

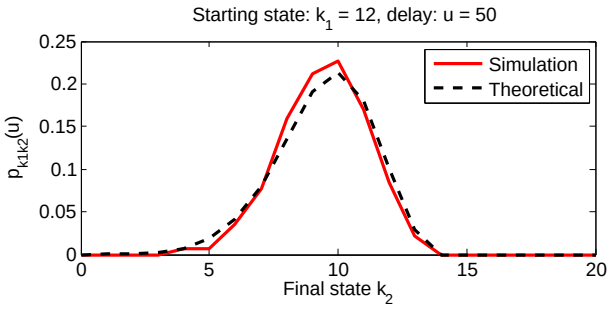

Fig. 3. Comparison of transition probabilities of surface traffic for one link, calculated using Monte Carlo simulations and using the theoretical approximation.

link will be randomly distributed at the time of delay assignment. A good approximation can be obtained by one of two methods. The most straightforward solution is to carry out Monte Carlo simulations. Alternatively, approximate transition probabilities can be calculated using the expected time-to-go ζ_k for each aircraft. The memoryless property of the component processes in the taxi time model allows ζ_k to be estimated for a given taxi path [10]. The probability $p_{k_1k_2}(u)$ can be calculated by assuming that the departure time of each aircraft is an exponential random variable with mean ζ_k. The resulting estimates are a good match with the simulated empirical distributions, as illustrated for a sample link in Fig. 3. Similar matches are seen for realistic multi-link paths in the BOS network. The theoretical approximation procedure is more useful than the Monte Carlo simulations, since it allows transition probabilities to be calculated quickly for arbitrary networks.

C. Dynamic programming formulation

As seen in Sec. III-B, the state of the system is given by $(\bar{N}, k)$, that is, the available gate capacity and the surface traffic level. The control variable is the pushback delay assigned to the first aircraft in the pushback buffer, which follows an FCFS order. State transition probabilities can be calculated using simulation or a theoretical approximation. However, while there is a finite number of buffer states (available gate capacity), the surface traffic level k can take any non-negative integer value, and the set of possible control inputs is the entire positive real line. To be able to compute optimal control values, the problem is reduced to a finite size by limiting the maximum traffic level to a value $k_{\max}$. The set of control variables is also discretized and restricted to a set $\mathbb{U}$. Thus the state space is limited to $\bar{N} \in \{0, 1, \ldots, N_{\max}\}$ and $k \in \{0, 1, \ldots, k_{\max}\}$, while the allowed control values are limited to the finite discrete set $\mathbb{U}$. The elements of $\mathbb{U}$ can be defined by the granularity required and any additional user preferences, such as the maximum gate delay acceptable to airlines. It is important to note that the traffic level is assumed to be bounded for policy calculation only, and not for the simulations presented in Sec. IV-C.

The costs in the current formulation are incurred by individual aircraft and by the airport. The cost for each aircraft is its fuel burn on the ground, which is proportional to its taxi time [1]. From Eq. (1), this component is $\eta + k\frac{X}{\mu}$. Since η is a constant, it can be omitted from the cost function. A second component of cost is incurred by the airport, as a result of the loss of throughput compared to the theoretical maximum. Since the cost function is in time units, this throughput loss is represented by the difference between the actual and the theoretical minimum inter-departure times. From Eq. (2), the penalty for throughput loss is given by $\Delta t_k - \Delta t_\infty = \frac{\eta}{k}$. This term is negligible for moderate to large traffic levels, but heavily penalizes low traffic levels. As a consequence, it drives the system away from states in which the runway may remain unused due to unavailability of departing aircraft. Finally, the pushback delay itself is part of the cost function. It corresponds to fuel consumption by auxiliary ground-based power sources at the gate. This cost component is thus proportional to the delay u. In summary, the expected per-stage cost for the system is given by

$$g(\theta, u) = g(\bar{N}, k, u) = \mathbb{E}\left[k_p(u)\,\frac{X}{\mu} + c_1\,\frac{\eta}{k_p(u)}\right] + c_2\,u.$$

The quantity $k_p(u)$ is the projected traffic level after time u and is characterized by $p_{k_1k_2}(u)$. The constants c_1 and c_2 are weights placed on the throughput loss and pushback delay, respectively. Finally, an *overflow tolerance* γ is defined, which is the maximum allowable probability of exceeding gate capacity given the current gate availability $\bar{N}$ and delay u. For the single-link case, γ corresponds to the probability of there being more than $\bar{N}$ arrivals to the buffer in time u. In physical terms, an overflow corresponds to an arrival waiting for a gate to be vacated by a departing aircraft. The value of γ sets an upper bound on the maximum delay assigned at any current state θ_1, thus defining a feasible subspace $\mathbb{U}(\theta_1)$ of the complete control space $\mathbb{U}$.

Once the stage cost and overflow tolerance are defined and the state transition probabilities are known, it is possible to solve for the optimal pushback delay at each state. The operating period for an airport lasts for the entire day, while the length of assigned pushback delays is expected to be a few minutes. Therefore, an infinite horizon formulation is the most suited to the current system. For computational simplicity, future costs are discounted by a factor $\alpha = 0.99$. By keeping the discount factor close to 1, sufficient importance is given to future states of the airport. As shown in [12], solving the set of Bellman equations in Eq. (4) yields the optimal policies $u(\theta_1)$ and costs $J(\theta_1)$:

$$J(\theta_1) = \min_{u \in \mathbb{U}(\theta_1)} \left(g(\theta_1, u) + \alpha \sum_{\theta_2} p_{\theta_1\theta_2}(u)\, J(\theta_2) \right). \quad (4)$$

D. Results for single-link formulation

The system of equations defined by Eq. (4) can be solved using the method of policy iteration, since this ensures termination in a finite number of steps [13]. Fig. 4 shows the resulting optimal policies for a given set of link and policy

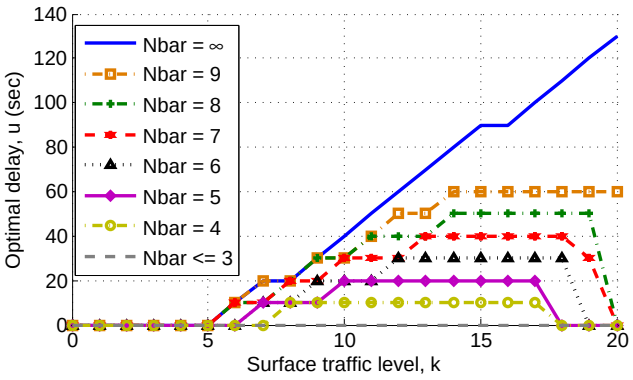

Fig. 4. Optimal policies for a single link. The link parameters are $\eta = 55$, $X = 0.2$, $\mu = 0.02$, $\beta = 0.06$, and $N_{\max} = 10$. The policy parameters are $\alpha = 0.99$, $c_1 = 5$, $c_2 = 0.9$ and $\gamma = 0.05$. Policy parameters are $k_{\max} = 20$ and $\mathbb{U} = \{0, 10, \ldots, 180\}$.

parameters. The value of β corresponds to an average inter-arrival time at the gates of 16.6 sec. As described in Sec. III-C, c_1 and c_2 weigh taxi time and fuel against throughput loss and pushback delay respectively. In this case, it is assumed that $c_1 = 5$ and $c_2 = 0.9$. An intuitive understanding of their effects can be gained by considering the tradeoff at some specific traffic level, say $\mathbb{E}[k_p(u)] = 10$. Changing this value to $\mathbb{E}[k_p(u)] = 11$ would increase the expected taxi time cost by $\frac{X}{\mu} = 10$, and reduce the expected throughput cost by $c_1\eta\left(\frac{1}{10} - \frac{1}{11}\right) = 2.5$. Thus the total additional cost would be 7.5 units, which would have to be offset by an 8.3 sec reduction in pushback delay. Note that this tradeoff is only approximate, since the transition probabilities also depend on the available gate capacity $\bar{N}$, and because the algorithm considers future costs as well.

In Fig. 4, the x-axis denotes the surface traffic level at the time of delay calculation, while the different curves denote the current availability of gates. The y-axis shows the optimal delay to be assigned to each $(\bar{N}, k)$ combination. The uppermost solid curve shows the policy calculated without consideration for the number of available gates, which is equivalent to an infinite gate capacity. A comparison of this curve with the other ones shows the effect of the overflow tolerance γ. As the buffer becomes full, the assigned delays decrease to the point where all aircraft are released immediately for $\bar{N} \leq 3$. Note that this does not guarantee that gate capacity will never be exceeded; for example, closely spaced arrivals during a delay assigned at some state with $\bar{N} \geq 4$ may still lead to an early pushback from the buffer.

Fig. 5 shows a simulation of the single-link network, with the control policies from Fig. 4. The infinite gate capacity policy results in an overflow of gates during a large portion of the simulation. By contrast, the finite gate capacity policy admits buffer overflow only once, approximately 6250 sec from the start of the simulation. When there is sufficient available gate capacity, the average taxi times for both policies are comparable. The finite gate capacity policy achieves the same benefits as the infinite capacity policy for moderate departure demand. When demand is high, the finite capacity policy allows some deterioration in taxi time performance in exchange for smooth operations. The apparent advantage of the infinite capacity policy in terms of taxi times under high demand, is an artifact of the simulation procedure. As explained earlier, frequent gate conflicts result in operational difficulties that lead to large delays that are not simulated here.

IV. Airport Network Control Strategy

While the single-link case is useful for simple networks, airports typically have complex layouts as well as several terminals (source nodes). In this section, the control strategy developed in Sec. III is extended to the full airport network with some modifications.

A. Configuration-specific network model

The network model for an airport is composed of several interconnecting links, as explained in Sec. II. Since most airports typically use only one or two departure runways at a time, only a subset of these links are active simultaneously. For example, when BOS is using Runway 27 for departures, only the highlighted part of the network in Fig. 1 is active. There are multiple pushback buffers in the network model, one corresponding to each source node (airport terminal). There are four such nodes at BOS (labeled 1, 2, 3 and 8), with their capacities defined by the gate capacity at each terminal of the airport. The runway (sink node) is labeled node 6. Since aircraft do not taxi in circular paths, the configuration-specific graphs are directed. There are only a few unique paths from each source node to the sink, which are assumed to be known *a priori* as described earlier.

The gate capacities of the four terminals are assumed to be 25, 20, 25 and 20 aircraft, which reflects the actual BOS gate capacity [14]. For policy calculation, the total arrival rate to these buffers is assumed to be $\beta = 0.02$, or an average of one arrival every 50 seconds. The terminal-specific arrival rates

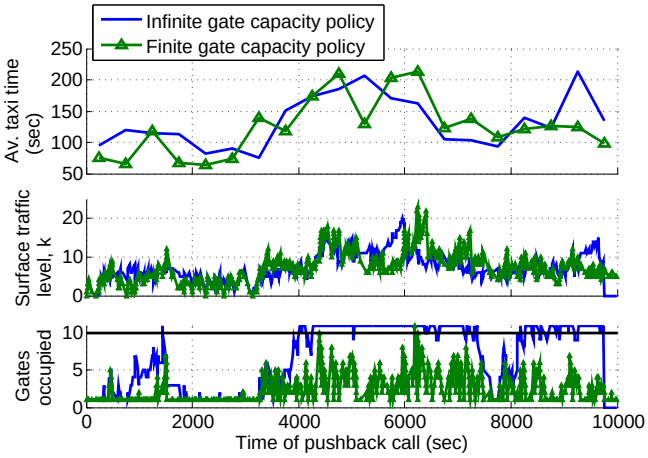

Fig. 5. Simulation of two control policies applied to a single-link network. The link parameters are $\eta = 55$, $X = 0.2$, $\mu = 0.02$, $\beta = 0.06$, and $N_{\max} = 10$. The policy parameters are $\alpha = 0.99$, $c_1 = 5$, $c_2 = 0.9$ and $\gamma = 0.05$. Policy parameters are $k_{\max} = 20$ and $\mathbb{U} = \{0, 10, \ldots, 180\}$.

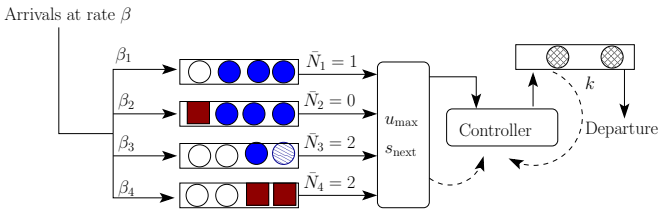

Fig. 6. Illustration of the multiple terminal model. The maximum delay corresponding to the combination of the four gate occupancy levels is $u_{\max}$. The hashed circle denotes the aircraft that is scheduled to push back next, and this source information is encoded in s_{next}. As before, the solid circles are the active aircraft in the pushback buffers, and the solid squares denote aircraft that are inactive (at their gates, but not ready to pushback). Aircraft that are actively taxiing are denoted by double-hashed circles.

β_i are assumed to be proportional to their gate capacities with $\sum \beta_i = \beta$. Aircraft that arrive at their gates are assumed to be inactive for the duration of their turnaround process, after which are ready to pushback.

B. State aggregation procedure

It is possible, in theory, to extend the single-link policy calculation procedure described in Sec. III-B to the calculation of optimal policies for the full network model. As shown in Fig. 6, the Poisson arrival process with rate β is split into four Poisson processes to the various terminals, each with rate β_i. Changes in gate occupancy levels are independent of changes in the surface traffic level, thus maintaining the decoupled nature of Eq. (3). However, the size of the resulting problem is very large, for two main reasons. Firstly, the optimal policy can vary depending on which source the next aircraft is leaving from. Additionally, with the given gate capacities, assuming that the maximum modeled traffic level is $k_{\max} = 25$ and that the control input set is $\mathbb{U} = \{0, 60, \ldots, 300\}$, the system has 31 million possible states. Solving the exact dynamic programming problem for a realistic airport model is therefore computationally infeasible.

Instead, state aggregation can be used to reduce the size of the problem [13]. Note that the combined effect of all four buffer states is the imposition of a constraint on the maximum delay that can be assigned. Using the maximum delay value $u_{\max}$ instead of the buffer states for policy calculation significantly reduces the size of the problem. In the BOS example being considered in this paper, the number of states decreases from 31 million to 624, corresponding to the product of 4 source nodes, 26 traffic levels and 6 choices for assigned delay. The state definition for policy calculation is now $(s_{\text{next}}, k, u_{\max})$, where s_{next} is the source corresponding to the next aircraft cleared for pushback. Sources for future states are assumed to be stochastic, with probabilities proportional to the gate capacities. Note that each gate availabilty state $[\bar{N}_1, \bar{N}_2, \bar{N}_3, \bar{N}_4]$ defines a unique $u_{\max}$, but the mapping is not unique in the opposite direction. Pushbacks are assumed to be First-Come-First-Served across all sources, except when one of the buffers exceeds its capacity, and a pushback is immediately released from that particular buffer. The aggregate formulation can be solved

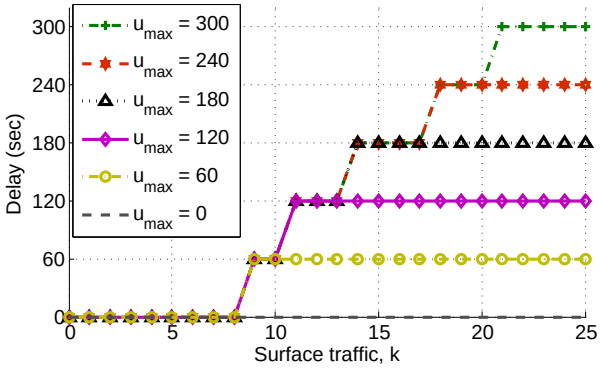

Fig. 7. Optimal policies for Boston Logan, when departures take place from Runway 27.

using the corresponding Bellman equations, analogous to the procedure described in Sec. III-C.

C. Results for airport network model formulation

The optimal delays calculated using the complete BOS network model for departures from Runway 27 are shown in Fig. 7. The policy illustrated in the figure is only for the departures leaving from source node 1. Similar policies are calculated for all four source nodes, and exhibit a similar staircase structure, as was previously seen in Fig. 4. The main difference is that in the complete network case, the different curves correspond to different $u_{\max}$ values, each of which encompasses several thousand buffer states. Since all the $u_{\max}$ curves level off before $k_{\max} = 25$, it is assumed that the corresponding maximum allowable delay is assigned to all traffic levels above $k_{\max}$.

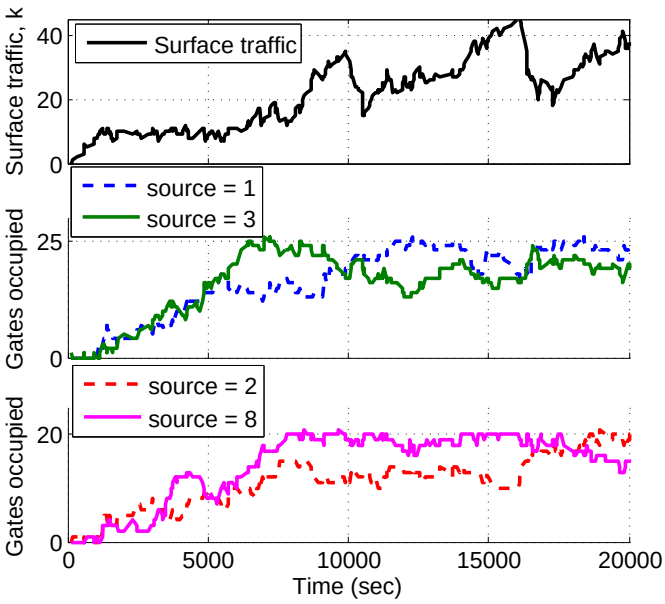

Fig. 8. Simulation of BOS operations for the runway configuration with departures from Runway 27. Pushback delays are calculated using the dynamic programming formulation proposed in Sec. IV-B.

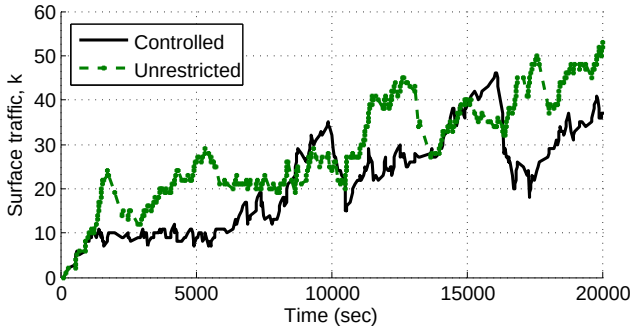

Fig. 9. Comparison of traffic levels with unrestricted pushbacks and the proposed control strategy.

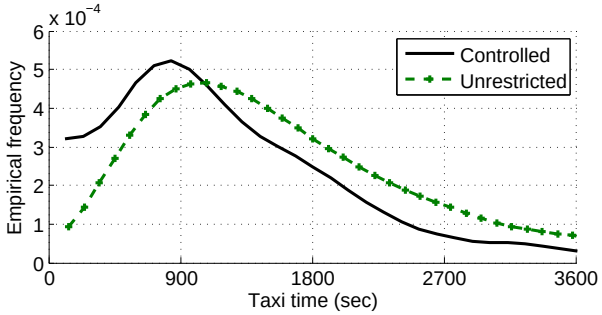

Fig. 10. Empirical distribution of taxi times under the two pushback strategies. The proposed control strategy is seen to result in significantly lower taxi times.

A simulation of the full departure process using the optimal policies calculated above is shown in Fig. 8. Note that the arrival rate $\beta = 0.02$ is very high considering the usual operational characteristics of BOS. However, this somewhat unrealistic level of demand emphasizes the differences between the proposed strategy and current control procedures. The simulation parameters are the same as those used previously, and the turnaround times are drawn from a uniform distribution that ranges from 30 to 45 min [15].

Fig. 8 (top) shows the variation of the surface traffic level with time. The middle plot shows the gate occupancy levels for the two larger terminals, while the bottom plot shows the gate occupancies for the smaller terminals. Two clear traffic peaks are seen in the simulation. In this example, both peaks are caused by the buffer corresponding to node 8 becoming full, thus necessitating pushbacks in rapid succession. There are 7 cases of buffer overflow amongst the 450 aircraft that pushed back during this period, which is reasonable compared to the overflow tolerance value (γ) of 5%.

Fig. 9 compares the results of the proposed control strategy with the current protocol, which is to release each aircraft as soon as it is ready. Aircraft called ready for pushback at the same times in both simulations. It is seen that for most of the simulation period, the traffic levels are higher in the case of unrestricted pushbacks. Fig. 10 shows that the corresponding taxi times are also higher. In this simulation, the proposed control strategy reduced taxi-out times by an average of 565 sec per aircraft.

V. Conclusions

This paper proposed a strategy for controlling pushbacks at an airport, that explicitly accounted for practical constraints such as gate availability. The objectives of the proposed optimization formulation were reduced taxi times, fuel burn and flight delays, with a limited impact on airport throughput. A realistic model of the airport surface, based on actual surface surveillance data, was used for taxi time prediction and simulation, and optimal control inputs were calculated using dynamic programming. The proposed strategy was shown through simulations to significantly reduce both taxi-out times and situations in which arrivals are delayed waiting for a gate. By generating a lookup table for pushback delays based on easily observable quantities such as the surface traffic level and gate occupancy, this method can be easily implemented at airports without requiring significant procedural modifications.

References

[1] H. Khadilkar and H. Balakrishnan, "Estimation of aircraft taxi-out fuel burn using Flight Data Recorder archives," in *AIAA Guidance, Navigation, and Control Conference*, Portland, OR, August 2011.

[2] I. Simaiakis, H. Khadilkar, H. Balakrishnan, T. G. Reynolds, R. J. Hansman, B. Reilly, and S. Urlass, "Demonstration of reduced airport congestion through pushback rate control," in *USA/Europe Air Traffic Management Research and Development Seminar*, Berlin, Germany, June 2011.

[3] E. Feron, R. J. Hansman, A. R. Odoni, R. Cots, B. Delcaire, W. Hall, H. Idris, A. Muharremoglu, and N. Pujet, "The departure planner: A conceptual discussion," White paper, MIT, International Center for Air Transportation, December 1997.

[4] H. Idris, B. Delcaire, I. Anagnostakis, W. Hall, N. Pujet, E. Feron, R. J. Hansman, J.-P. Clarke, and A. Odoni, "Identification of flow constraint and control points in departure operations at airport systems," in *AIAA Guidance, Navigation and Control Conference*, August 1998.

[5] N. Pujet, B. Delcaire, and E. Feron, "Input-output modeling and control of the departure process of congested airports," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Portland, OR, 1999, pp. 1835–1852.

[6] P. Burgain, "On the control of aircraft departure operations," Ph.D. dissertation, Georgia Institute of Technology, November 2010.

[7] C. Brinton, J. Krozel, B. Capozzi, and S. Atkins, "Improved taxi prediction algorithms for the surface management system," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Monterey, CA, August 2002.

[8] H. Balakrishnan and Y. Jung, "A framework for coordinated surface operations planning at Dallas-Fort Worth International Airport," in *AIAA Guidance, Navigation, and Control Conference*, Hilton Head, NC, August 2007.

[9] H. Lee, I. Simaiakis, and H. Balakrishnan, "A comparison of aircraft trajectory-based and aggregate queue-based control of airport taxi processes," in *Digital Avionics Systems Conference*, Salt Lake City, UT, October 2010.

[10] H. Khadilkar and H. Balakrishnan, "Network congestion control of airport surface operations," *to appear in Journal of Guidance, Control and Dynamics*.

[11] ——, "Metrics to evaluate airport operational performance using surface surveillance data," *to appear in Air Traffic Control Quarterly*.

[12] D. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 1*. MA: Athena Scientific, 2005.

[13] ——, *Dynamic Programming and Optimal Control, Vol. 2*. MA: Athena Scientific, 2007.

[14] "About Logan," http://www.massport.com/logan-airport/about-logan/Pages/Default.aspx, retrieved September 2012.

[15] C.-L. Wu and R. Caves, "Modelling and optimization of aircraft turnaround time at an airport," *Transportation Planning and Technology*, vol. 27, no. 1, pp. 47–66, 2004.