Control of Aircraft Pushbacks at an Airport using a Dynamic Programming Formulation

Journal:	*AIAA GNC/AFM/MST/ASC 2012*
Manuscript ID:	Draft
luMeetingID:	2423
Date Submitted by the Author:	n/a
Contact Author:	Khadilkar, Harshad; Massachusetts Institute of Technology, Aeronautics and Astronautics

Control of Aircraft Pushbacks at an Airport using a Dynamic Programming Formulation

Harshad Khadilkar* and Hamsa Balakrishnan†

Massachusetts Institute of Technology, Cambridge, MA 02139, USA

This paper describes a dynamic programming formulation of the airport surface traffic management problem. Movement of aircraft is modeled as the flow of traffic on a network, with stochastic link travel times. This is followed by an algorithm for controlling entry of aircraft into the taxiway system at an airport. Finally, two realistic variations of the formulation are presented - variation of parameters and finite buffer capacity. Optimal control policies for all cases are calculated using policy iteration, with delay mitigation and aircraft fuel burn reduction as explicit objectives.

I. Introduction

A. Motivation

Various studies[1–4] have proposed delaying aircraft at the gate as a good strategy for reducing taxi times and surface fuel burn when the airport is experiencing congestion. Aircraft waiting at the gate typically have not started their engines, and therefore do not burn as much fuel as aircraft in a runway departure queue. When assigned appropriate delays, aircraft encounter much lower congestion on the surface, thus also reducing their taxi times. However, these studies first aim to reduce or limit airport surface congestion, and then evaluate the incidental benefits received in terms of fuel burn and taxi time savings.[5] A more direct way of achieving these benefits would be to include fuel burn and taxi times as the objectives in an optimization problem. This would not only give more insight into the interactions between aircraft and the airport infrastructure, but would also allow for less conservative control algorithms. In this paper, we propose to address these issues using dynamic programming. We use an infinite-horizon discounted cost formulation to model the operations at a busy airport during periods of peak demand. To be compatible with current levels of air traffic control technologies, we only consider the problem of determining the optimal time of entry into the network, and assume that aircraft routes are fixed.

B. Definitions

Some terms related to surface operations at airports are defined here for the sake of clarity.

1. Gate: Parking bay for aircraft attached to the airport terminal, where passengers board and deplane.
2. Pushback: 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.
3. Pushback delay: An instruction given to an aircraft, delaying the start of its pushback process. The aircraft is supposed to wait at the gate until it is cleared for pushback.
4. Pushback buffer: All aircraft that have completed their departure preparations and have called the air traffic controller for permission to pushback, but have not been cleared for pushback.

*Graduate Student, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139. harshadk@mit.edu. AIAA Student Member.

†Associate Professor, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139. hamsa@mit.edu. AIAA Senior Member.

II. Problem Formulation

A. System Description

In this paper, we use a network model for taxi times at an airport, that was developed in prior work.[6,7] We model the airport surface as a network, consisting of the aircraft gates (*source nodes*), major taxiways (*links*), and runways (*sink nodes*). For example, the model for Boston Logan International Airport (BOS) is shown in Figure 1. Gates corresponding to each of the four airport terminals at BOS are associated with nodes 1, 2, 3 and 8. The runways are at nodes 6, 7, 10, 11, 12, 14 and 15. There are some intermediate nodes, such as 4 or 5, that are neither sources nor sinks, but are intersections of major taxiways.

Figure 1. Network model of Boston (BOS) airport. Nodes are marked using white boxes. The links highlighted in light blue constitute the configuration-specific network for departures from Runway 27.

A set of random processes is used to model the taxi operations of aircraft on the surface. The taxi time of each aircraft, on each link in the network, is the sum of two random variables: (i) the unimpeded taxi time, and (ii) the time it spends stopped on the surface. The expected time for which an aircraft is stopped on the surface (and hence the expected total taxi time), increases with the number of aircraft already on the surface when the current aircraft leaves its gate. In this preliminary work, we assume that the entire airport can be modeled as a single link, that is, *there is a single link with the airport gates at the source, and the runway at the sink*. The concepts presented here can be extended to the full network quite easily.

The taxi-time of an aircraft is modeled as

$$t_t = t_u + \sum_{i=1}^{N_s} t_{s,i}. \tag{1}$$

- $t_u > 0$ is the unimpeded travel time, an Erlang random variable with order n and rate λ.
- $N_s \in \{0, 1, 2, ...\}$ is the number of stops made by the aircraft, modeled as a geometric random variable with parameter $p_k \in [0, 1)$, where k is the current number of aircraft on the surface. If $N_s = 0$, then $t_t = t_u$.
- $t_{s,i} > 0$ is the stationary time corresponding to the i^{th} stop, modeled as an exponential random variable with rate $\mu > 0$. Each $t_{s,i}$ is assumed independent of t_u, $t_{s,j\neq i}$ and N_s, and identically distributed (*iid*) for a given link.
- The probability of stopping (geometric with parameter p_k), is defined in such a way, that the expected

taxi time on each link increases by a fixed amount for every aircraft added to the network.

$$\begin{aligned}\mathbb{E}[t_t|k] &= \underbrace{\mathbb{E}[t_u|k]}_{\text{Independent of } k} + \mathbb{E}\left[\sum_{i=1}^{N_s} t_{s,i}\middle| k\right] \\ &= \frac{n}{\lambda} + \underbrace{\frac{p_k}{1-p_k}\frac{1}{\mu}}_{\text{Linear in } k} \\ \mathbb{E}[t_t|k] &= \frac{n}{\lambda} + \left[\frac{p_0}{1-p_0} + k\,X\right]\frac{1}{\mu} = \eta + \frac{k\,X}{\mu}. \end{aligned} \tag{2}$$

Here, X is a constant that controls the rate at which expected taxi time increases with k, the number of aircraft in the network. The length of each stop is assumed to be independent of previous stops. Note that η does not depend on k, and is only a function of n, λ, p_0 and μ.

The more time that an aircraft spends taxiing, the greater its fuel cost.[8,9] On the other hand, delay assigned to an aircraft at the gate also incurs some cost, as excessive gate delays will result in takeoff delays and in conflicts with other aircraft that need to park at the same gate. Thus, there is a tradeoff between the additional fuel cost due to long taxi times, and the cost of delaying entry into the network.

B. Control Problem

In this section, we develop a dynamic programming formulation of the control strategy for the system described above. The *state* of the system is the number of aircraft on the surface (k), just after each pushback. Technically, $k \in \{0, 1, ...\}$ is a countably infinite set. However, in this formulation we use hard state aggregation to combine all states beyond a certain threshold k_{max}, into a single state denoted by k_{agg}. Therefore, the set of states is now a finite set, with $k \in \{0, 1, ..., k_{max}, k_{agg}\}$. The *control input* (pushback delay) is calculated whenever an aircraft calls ready to push from its gate. This delay, $u \geq 0$ is a function of the current level of traffic on the surface. The aircraft receiving the delay will pushback from its gate after a further time u has elapsed. For computational simplicity, the available set of controls $\mathbb{U}$ is discretized and bounded so that $u \in \{0, u_1, ..., u_m\}$. *State transitions* are defined to occur at time instants when a new aircraft pushes back. Since future states depend on the random processes governing the taxi times of aircraft already on the surface, destination states are random. There is therefore a state transition probability associated with each proposed value of gate delay, $p_{kj}(u)$. We note that our definition of state transition means that the time increment to each transition is not the same. However, since costs are associated with discrete aircraft and are not time-averaged, we are interested in the evolution of the discrete states of the system, rather than the transition times. *Stage costs* are composed of two variables. The gate delay, u, is itself part of the stage cost. In addition, we have the fuel consumption of the aircraft. We assume that this is proportional to its taxi time[9] t_t (in this case, the link travel time). We use a proportionality constant, a, that weighs the gate delay u against the taxi time t_t. The stage cost in the linear case is therefore

$$g(k, u) = a\,u + \mathbb{E}[t_t|k, u].$$

Note that t_t is a random variable, with a known distribution that depends on k. In Section III-C, we also consider a quadratic stage cost. Finally, some additional points to note regarding this problem are:

- The service is assumed to be First-Come-First-Served (FCFS). Aircraft pushback in the same order as they call ready for pushback. However, once out on the link, aircraft may overtake each other. This is quite realistic with respect to current air traffic control policies, which are FCFS in clearance delivery, but the sequences may change on the way to the runway because of differing taxi speeds, distances to the runway, etc.
- In the initial parts of this paper, we assume the existence of an infinite buffer of aircraft always ready to pushback. In other words, we assume that another aircraft is always waiting when the current aircraft leaves its gate. While this is a good approximation of high departure demand periods at a busy airport, a more general model is developed and illustrated in Section IV-B.

- Gate delay is assigned only once, namely, when the aircraft is next in queue for pushback. The aircraft leaves its gate at this assigned time, and cannot be further delayed (or be asked to leave earlier than its assigned time).

C. Problem Statement

The infinite horizon, discounted cost, dynamic programming problem can be formulated using the set of Bellman's equations,[10]

$$J(k) = \min_{u \in \mathbb{U}} \left(a\,u + \mathbb{E}[t_t|k,u] + \alpha \sum_{j=0}^{k_{agg}} p_{kj}(u)\,J(j) \right). \tag{3}$$

Here, $J(k)$ is the infinite-horizon expected cost-to-go for a fixed policy $\mathcal{U}$ starting from the state k, α is the discount factor, and $\mathbb{U}$ is the set of allowed controls. The time instants of decision (and hence the state transitions) are assumed to be the times of each pushback. As stated before, u is discretized and bounded, and all states $k > k_{max}$ are aggregated into k_{agg}. The expected taxi times, conditional on $u = \mathcal{U}(k)$, can be calculated using Equation (2) and the state transition probabilities, $p_{kj}(u)$.

III. Results

A. Estimation of Transition Probabilities

The actual number of states for this system is countably infinite. For the estimation of transition probabilities, we can aggregate all states above some threshold, into a single state. If this threshold is k_{max}, all transitions $k_1 \to k_2$ for $k_1, k_2 > k_{max}$ count as transitions $k_{agg} \to k_{agg}$. In addition, the control set is continuous as well as infinite. To handle this, we discretize the allowed controls into a finite set $\mathbb{U}$ for estimation purposes.

For the initial part of the problem, we used a hypothetical link with states from $k = 0$ to $k = 30$ and an additional state for all $k > 30$. Thus, there were a total of 32 states in the estimation procedure. The link parameters (for the model presented in Section II-A) were $n = 20$, $\lambda = 0.4$, $X = 0.4$, and $\mu = 0.04$. The transition probabilities were calculated by running simulations with randomized policies. The control set at any state was $\mathbb{U} = \{0, 10, 20, 30, 40, 50, 60, 70, 80\}$ seconds. The 10 second step size was chosen to be close to the expected interdeparture time between two aircraft, when the system was at the intermediate state $k = 15$. Each time the maximum state was exceeded, the simulation was restarted from $k = 0$ (empty link). This was done to ensure sufficient exploration of all state-control pairs.

B. Results for Linear Costs

As defined earlier for the case of linear costs, the single stage costs are,

$$g(k,u) = a\,u + \mathbb{E}[t_t|k,u].$$

For the finite-state, finite-control case, we can solve for the infinite-horizon discounted costs by direct matrix inversion. For a fixed policy $\mathcal{U}$, the general form of Equation (3) can be rewritten using Equation (2) as,

$$J(k) = a\,\mathcal{U}(k) + \sum_{j=1}^{k_{agg}} p_{kj}(\mathcal{U}(k)) \left[\eta + \frac{(j-1)\,X}{\mu} + \alpha\,J(j) \right].$$

Note that for the infinite buffer case, the summation starts at $j = 1$ instead of $j = 0$, since there is always an aircraft waiting at the instant after every pushback, and so decisions are always taken with $k > 0$ aircraft on the link. Secondly, $\mathcal{U}(k)$, the control for state k defined by stationary policy $\mathcal{U}$, is the assigned pushback delay. Hence, $a\mathcal{U}(k)$ is the deterministic part of the stage cost. The expected taxi time for the current aircraft, and the future costs, depend on the state of the system at pushback, and are therefore included in the expectation. Thirdly, a transition $k \to j$ (where j is the state just after pushback) implies that the number of active aircraft at pushback was $(j-1)$, which then became j when the current aircraft pushed back. Hence the expected taxi time term has a coefficient of $(j-1)$. Finally, the exception to this formula is the aggregate state (k_{agg}), which handles all $k > k_{agg}$. In this case, the expected taxi time and future costs need to be adjusted appropriately, since an aircraft may push back in state k_{agg}, but the airport will also incur future costs at state k_{agg}.

The optimal policy for this example, produced by policy iteration, is to push all aircraft immediately for $k \leq 3$, and to delay all aircraft for the maximum 80 seconds for $k > 3$. The policy convergence is shown in Figures 2 and 3. These figures depict the convergence of costs associated with successive fixed policies, starting from two extreme initial policies. This 'bang-bang' behavior is due to the fact that the expected taxi time decreases linearly with pushback delay. Therefore, if the rate of decrease exceeds the rate of increase of $a\,u$, it is optimal to delay all aircraft for as long as possible. If it is lower than the rate of increase of $a\,u$, it is optimal to release all aircraft immediately.

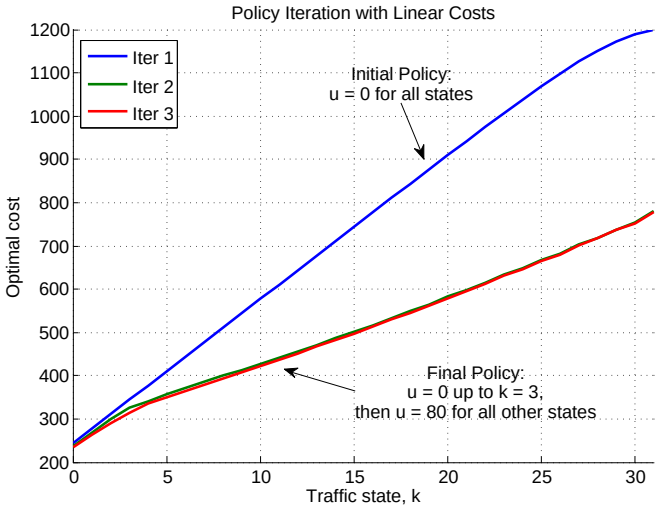

Figure 2. Policy iteration, starting with an initial policy of pushing back everyone immediately.

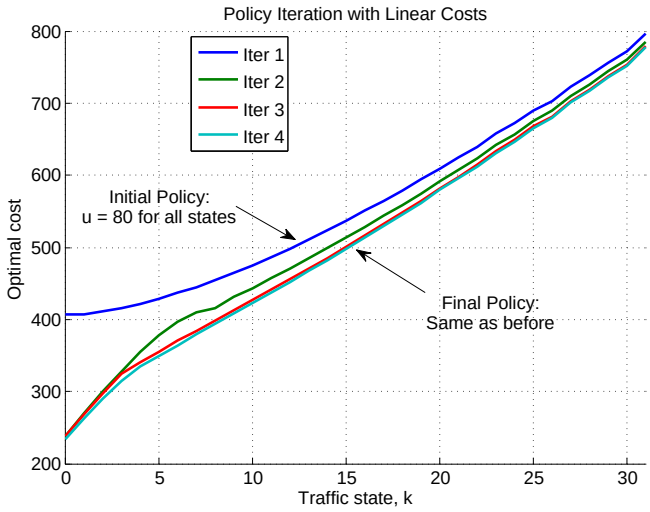

Figure 3. Policy iteration, starting with an initial policy of holding everyone for the maximum 80 seconds. Note the different scale on the y-axis, as compared to Figure 2.

C. Solution with Quadratic Costs

The result of implementing the optimal policy for linear costs (Section III-B) is that there are large fluctuations in surface traffic. The traffic level increases rapidly to the point where pushbacks are stopped altogether, then falls back down to a small number, and the process repeats. This behavior is not desirable

from a practical standpoint. To get smoother policies, we add a small quadratic term to the stage-wise costs. The new cost function is given by,

$$C_q = t_t + a\,u + b\,u^2.$$

We use a value of b that is small compared to a. The derivative of the cost is given below. We know, from Section III-B, that the first term is linear with a negative slope, and that the value of the slope is a function of the state k. Therefore, we are guaranteed a finite optimal value for u. If the unconstrained optimum for u is negative, we use $u = 0$.

$$\frac{dC_q}{du} = \frac{dt_t}{du} + a + 2\,b\,u$$

In this case, the optimal policies that we get are smoother than before. The results for $\alpha = 0.7$ are shown in Figures 4 and 5.

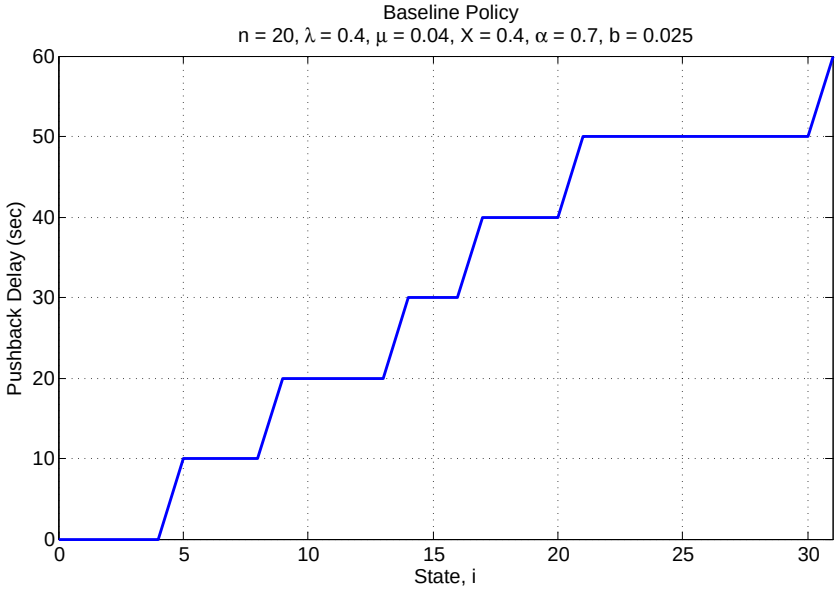

Figure 4. Optimal infinite-horizon policy for quadratic costs, with $\alpha = 0.7$.

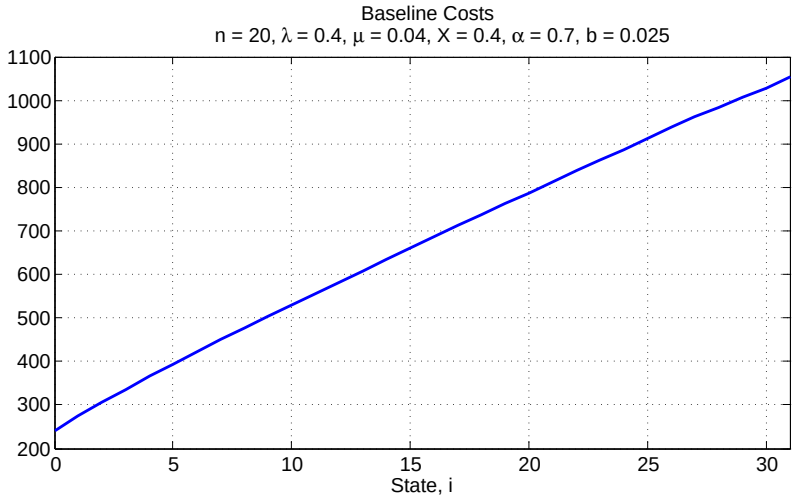

Figure 5. Infinite-horizon discounted costs for quadratic case, with $\alpha = 0.7$.

IV. Variations of the Original Formulation

We now consider two variations of the formulation described above, both of which are relevant from the perspective of the real system.

1. *Parameter variation:* In this case, we assume that the value of η is not fixed, but can vary over a certain range. This adaptation reflects situations in which the unimpeded taxi times of aircraft vary over a given day because of weather, surface conditions, time of the day, and so on.

2. *Limited buffer capacity:* In this variant, we allow the buffer that holds aircraft to be of limited size. This reflects the fact that the number of gates at an airport are limited, and when these are filled, aircraft must be released immediately to accommodate new arrivals.

A. Rollout Algorithm for Varying Parameters

We assume that the value of η, the part of expected taxi time that does not depend on k, is piecewise constant within the range $[0.2\eta_0, 1.8\eta_0]$. Here, η_0 is the nominal value of η. When simulations are being run, we start with a random value of η with repeated draws after certain time intervals. Parameter values for the underlying random processes are derived from the current value of η.

To implement the solution to this problem, we calculate the optimal baseline policy and the approximate infinite horizon discounted costs $J_{\eta_0}(i)$ based on η_0, the nominal value of η. In the rollout implementation of the solution, we estimate the current value of η online, using past travel times over the link. This estimated value, $\hat{\eta}$, is used for estimation of current single stage cost. The update equation for $\hat{\eta}$ is based on the error between current expected taxi time and the actual observed taxi times, and is given by

$$\hat{\eta}_{r+1} = \hat{\eta}_r + G\left(\hat{\eta}_r + k(r)\,\frac{X}{\mu} - t_{t,r}\right).$$

Here, r is the index of the last aircraft to depart from the link, $k(r)$ is the surface traffic level when it entered the network, $t_{t,r}$ is its actual taxi time, and G is a gain that weighs current observations against the estimate of $\hat{\eta}$. The control for the $(r+1)^{\text{th}}$ aircraft is found using the following equation:

$$u_{r+1} = \min_{u\in\mathbb{U}}\left[a\,u + b\,u^2 + \sum_j p_{kj}(u)\,\left(\hat{\eta}_{r+1} + (j-1)\,\frac{X}{\mu} + J_{\eta_0}(j)\right)\right].$$

In other words, we use the current parameter estimates for predicting single-stage costs, and the nominal values for predicting future costs. The results from this procedure are shown in Figure 6. A time window of 50,000 seconds was used for each simulation run. The figure shows that the rollout algorithm shows consistent improvement over the baseline policy, over 10 independent simulation runs. The variation of the estimate $\hat{\eta}$ and the actual value of η are compared in Figure 7 for one of the runs.

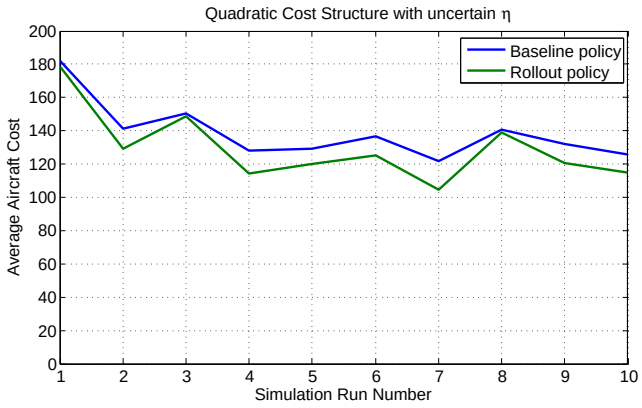

Figure 6. Average simulated aircraft per-stage cost with uncertain η, for ten simulation runs.

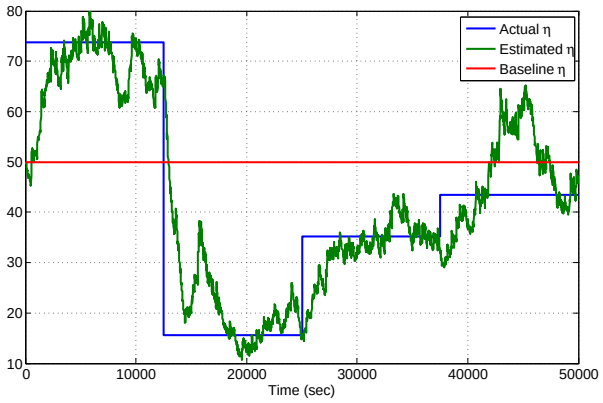

Figure 7. Online estimation of η for the last simulation run. The blue line indicates the parameter value used for generating taxi times. The green line indicates $\hat{\eta}$ as estimated from the generated taxi times.

B. Formulation with Finite Buffer Capacity

This model reflects the fact that there are limited gates at any airport, and delayed aircraft may have to be released early if another aircraft needs to use their gate. The assumptions made here are:

- Calls for pushback occur at a Poisson rate β. In this case, we used $\beta = 0.05$, which gives an average interval of 20 sec between successive calls for pushback.
- A maximum of $(N_{max} - 1)$ aircraft can be held at the gates. When the buffer level reaches N_{max}, the first aircraft must be released immediately.
- The taxi time model $(n, \lambda, \mu, X, p_0)$ is unaffected.

In this case, our state is of the form (N, k) where $N \in \{1, 2, \ldots, N_{max} - 1\}$ denotes the current buffer level, and $k \in \{0, 1, 2, \ldots\}$ is the number of aircraft on the link. The epochs are defined at the moment of each pushback. If the buffer is empty following a pushback, the epoch is defined by the arrival of the next call for pushback, from the Poisson process β. The possible types of transitions are shown in Figure 8. If k_{max} is the maximum modeled traffic level on the link, there are a total of $(N_{max} - 1)(k_{max} + 1)$ states in the full model. Compared to the infinite buffer capacity model, there will be many more possible transitions, with the number of state-control pairs increasing by a factor of $(N_{max} - 1)$. The issue of state exploration becomes more critical in the transition probability estimation procedure. However, the Bellman equations will still be of the same form, and in theory, the solution procedure does not require any modification. Figure 9 explains the process: A state transition happens at each pushback, and at that time, a delay is assigned to the next aircraft in the buffer. Delays for the remaining aircraft in the buffer have not yet been calculated.

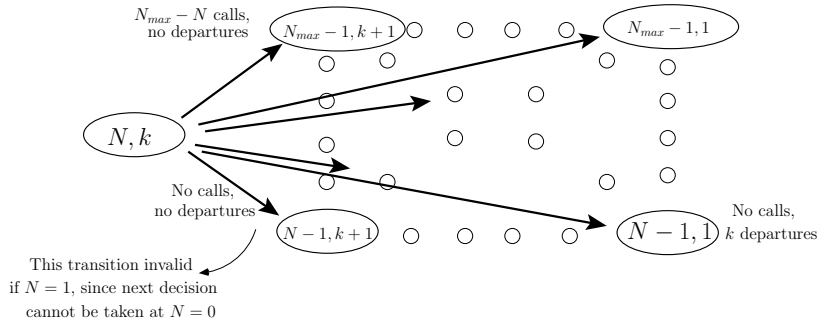

Figure 8. Possible state transitions for the limited-capacity buffer case.

The optimal policies for this case, calculated using policy iteration, show interesting characteristics. We would expect that as the buffer approaches maximum capacity, the policy would shift to letting aircraft

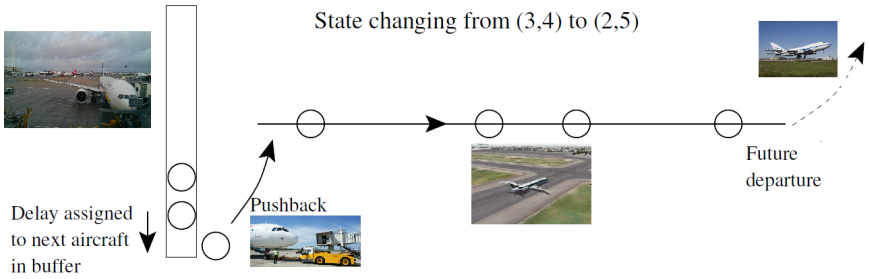

Figure 9. Depiction of the control process in the limited buffer case.

pushback earlier. This is confirmed by Figure 10. As N increases, the assigned gate delay becomes lower in order to avoid exceeding maximum capacity. For a fixed value of N, the assigned delay increases with k, just like before. Another interesting point to note relates to the curve for $N = 9$. At this point, there is no further space in the buffer, and the next call from the Poisson process (β) will force an early pushback. The next expected arrival time from the Poisson process is $\frac{1}{\beta} = 20$ seconds, which corresponds almost exactly to the assigned gate delay for $N = 9$. That is, the next pushback is scheduled at the same time that the buffer is expected to hit maximum capacity.

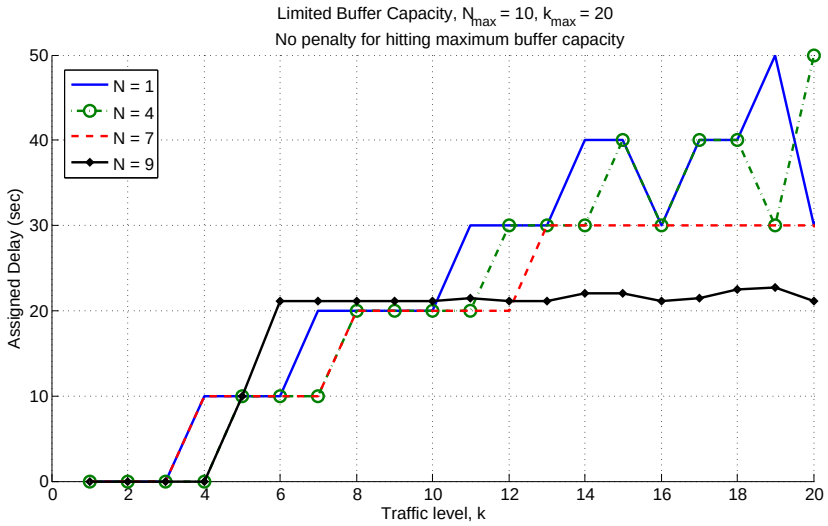

Figure 10. Optimal policy for the limited buffer case, without a penalty for forced pushbacks. Calls for pushback are assumed to be arriving at the Poisson rate $\beta = 0.05$.

If we include a penalty for exceeding maximum buffer capacity, the policies for the higher values of N become even more conservative, as shown in Figure 11. The optimal policies for the lower values of N are almost unchanged, while lower gate delays are assigned when the buffer is close to being full. For the boundary case of $N = 9$, no risk is taken at all, and the next aircraft is always released without delay.

V. Conclusions

This paper proposed a dynamic programming formulation to solve the airport congestion control problem. The advantage of this method was that we could directly optimize for the benefits of reduced fuel burn and taxi times, instead of an implicit traffic-limiting method where benefits are secondary outputs. We showed that defining an objective function consisting of pushback delay and taxi times, and using a weakly quadratic structure, gives reasonable control policies. In addition, we also modified the formulation to handle two realistic variations of the problem: (i) varying parameters, and (ii) limited-capacity buffers. In the first

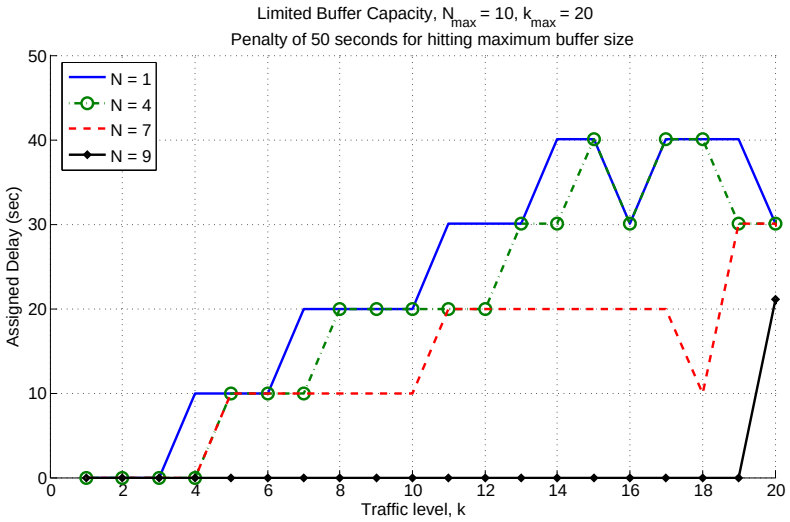

Figure 11. Optimal policy for the limited buffer case, with a penalty of 50 sec for forced pushbacks. Calls for pushback are assumed to be arriving at the Poisson rate $\beta = 0.05$.

case, using a rollout algorithm was shown to improve the performance of the baseline control policy. In the second case, the resulting control policies were shown to become more conservative as the buffer started approaching maximum capacity. This tendency was shown to be enhanced when a penalty for reaching maximum capacity was added.

References

[1]Feron, E., Hansman, R. J., Odoni, A. R., Cots, R., Delcaire, B., Hall, W., Idris, H., Muharremoglu, A., and Pujet, N., "The Departure Planner: A Conceptual Discussion," White paper, MIT, International Center for Air Transportation, December 1997.

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

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

[4]Burgain, P., *On the control of aircraft departure operations*, Ph.D. thesis, Georgia Institute of Technology, November 2010.

[5]Simaiakis, I., Khadilkar, H., Balakrishnan, H., Reynolds, T. G., Hansman, R. J., Reilly, B., and Urlass, S., "Demonstration of Reduced Airport Congestion Through Pushback Rate Control," *USA/Europe Air Traffic Management Research and Development Seminar*, Berlin, Germany, June 2011.

[6]Khadilkar, H., *Analysis and modeling of airport surface operations*, Master's thesis, Massachusetts Institute of Technology, June 2011.

[7]Khadilkar, H. and Balakrishnan, H., "A Network Congestion Control Approach to Airport Departure Management," *American Control Conference*, June 2012.

[8]Jung, Y., "Fuel Consumption and Emissions from Airport Taxi Operations," NASA Green Aviation Summit, 2010.

[9]Khadilkar, H. and Balakrishnan, H., "Estimation of aircraft fuel burn using flight data recorder archives," *AIAA Guidance, Navigation and Control Conference*, Portland, OR, August 2011.

[10]Bertsekas, D., *Dynamic Programming and Optimal Control, Volume I*, Athena Scientific, 3rd ed., 2005.