

Approximate Dynamic Programming for Networks: Fluid Models and Constraint Reduction

Michael H. Veatch
Department of Mathematics
Gordon College
mike.veatch@gordon.edu

August 19, 2009

Abstract

This paper demonstrates the feasibility of using approximate linear programming (ALP) to compute nearly optimal average cost for multiclass queueing network control problems of moderate size. The method requires only solving an LP and approximates the differential cost by a linear form. Quadratics, certain exponential functions, piece-wise quadratic functions from fluid models, and other approximating functions are used. The ALP is made more tractable for these three types of functions by algebraically reducing the constraints to a smaller equivalent set. On examples with two to six buffers, bounds on average cost were 12 to 40% below optimal using quadratic approximating functions; error was reduced to 2 to 21% by using about twice as many approximating functions and can be reduced further by systematically adding functions. Although the size of the LP is exponential in the number of buffers, examples with up to 17 buffers and quadratic approximation are solved. For a given level of accuracy, the method requires much less computation than standard value iteration. Policies are also constructed from the ALP.

1 Introduction

The increasing size, complexity, and flexibility of manufacturing processes, supply chains, communications, and computer systems have made them increasingly difficult to model and to operate efficiently. Recurrent themes seen in many industries are a large number of interacting processes and significant randomness due to customer demand which must be rapidly served, in addition to uncertainties in the service process. The natural modeling framework for these systems is a multiclass queueing network (MQNET). Even under the simplest assumptions of exponentially distributed service and interarrival times and linear holding costs, MQNET control problems are NP-hard so that we cannot hope to solve large problems exactly [22]. Standard policy iteration or value iteration are too computationally intensive to use even on moderate size problems, particularly in heavy traffic, due to the well-known “curse of dimensionality.”

This paper demonstrates the feasibility of using approximate linear programming (ALP) to approximate optimal average cost for MQNETs dramatically faster than exact dynamic programming (DP) methods.

A lower bound on optimal average cost could be very valuable for benchmarking the many heuristics for controlling these systems. The justification of these heuristic policies has been less than satisfactory. Many heuristic policies have been shown to be stable, but relatively little is known about suboptimality. Certain policies have been shown to be asymptotically optimal for the limiting Brownian control problem in heavy traffic or the fluid control problem, which essentially considers large buffer contents. However, asymptotic optimality is generally too loose a criteria for designing near-optimal policies.

The main challenge in applying any approximate DP method, including ALP, is selecting a compact but accurate class of approximating functions for the differential cost. The first contribution of this paper is identifying new approximating functions that are shown numerically to improve accuracy. More specifically, an improved trade-off between speed or size of the LP and accuracy is achieved. We begin with a quadratic approximation, which has been used by several researchers, and test its speed and accuracy. A natural refinement of the quadratic approximation is the “cost to drain” of the associated fluid model, which is either quadratic or piece-wise quadratic. Fluid cost has been used to initialize the value iteration algorithm [5]. In this paper we find and use these quadratic regions for a three-class example. We also use approximating functions with certain exponential terms and indicator functions that aggregate states based on an estimate of their importance to the approximation. These approximating functions are motivated by the problem structure and numerical experience.

The second challenge in solving the ALP is that it contains one constraint for every state-action pair, which is impractical for large networks. An open network with uncontrolled arrivals has an infinite state space; truncating by limiting buffer sizes results in an ALP that is too large. Without some method of reducing the number of constraints, the alternatives of approximate value iteration or approximate policy iteration may be preferable. The second major contribution of this paper is to provide new constraint reduction methods for piece-wise quadratic functions and certain exponential functions. We also use the constraint reduction method in [20] for quadratic functions. For each type of function, the number of constraints is still exponential in the number of buffers; however, the reduction extends considerably the size of problem for which the ALP can be solved. The ALP approach also has the simplicity of just solving an LP and the advantage of somewhat stronger theoretical results than for the other methods; see [7] and [25].

We also briefly addresses the *policies* associated with the differential cost approximations. The approximation architectures are compact enough that one could implement these policies. However, testing has shown that they do not always have good performance and may not even be stabilizing, as demonstrated in [6] for a related control problem involving a single queue. We report some cases where the ALP gives a useful policy. A final contribution involves upper bounds on average cost. An upper bound ALP for a specific policy is given in [20]. We propose solving the lower bound ALP and using the associated policy in the upper bound ALP. A second bound is derived that uses Bellman error.

Accuracy of the ALP bound was tested on networks with up to six buffers which could be solved using value iteration. In general, accuracy improves as more basis functions are added to the approximation. The quadratic approximation uses the smallest LP and gave an average cost 12 to 40% below optimal. When exponential or piece-wise quadratic approximations were added, roughly doubling the number of variables, accuracy improved to 2 to 21%. Adding indicator functions gradually achieved greater accuracy: for a two-class series queue with traffic intensity of 0.8, an error of less than 1% was achieved using 113 functions, including indicator functions for individual states. For a three-class reentrant line with a traffic intensity of

0.9, an error of 5% was achieved using 2309 functions. These LPs used dramatically fewer variables and less time than is needed to achieve the same accuracy with value iteration. Speed of the ALP with quadratic approximation was tested on larger examples, demonstrating that it can be solved on series lines with 17 buffers. Even larger multiclass networks could be solved, since they have fewer servers and control actions.

The ALP approach was originally proposed by [26]. It is applied to discounted network problems in [7] using quadratic value function approximations. They provide an error bound for the ALP value function. In particular, a suitable weighted norm of the error is bounded by the minimum of this error norm over all functions in the approximating class, multiplied by a constant that does not depend on problem size. Similar bounds are given on performance of the policy implied by the ALP value function. Instead of constraint reduction, they use importance sampling of constraints, which is shown to be probabilistically accurate in [8]. For average cost problems, two modifications of the ALP approach are proposed in [6] and [9]. Although the latter provides a performance bound, it is not clear how to apply it to networks. In [29], column generation methods are used to solve average cost ALPs more efficiently. Approximate dynamic programming is applied to a specific communication network—a crossbar switch—in [19] using “ridge” functions of a single variable and to inventory problems in [1] and [28] using linear functions. We tested separable approximations, i.e., functions of a single buffer size, on several MQNETs and found that they were not as accurate as any of the functions listed above—even quadratics. A major difference between our work and [19] is that we identify different approximating functions which give a better trade-off between speed and accuracy for a test suite of MQNETs.

The papers most closely related to ours are [20] and [21], which also consider average cost ALPs and piece-wise quadratic approximations. Using a different quadratic on each set of states defined by which buffers are empty, they reduce the constraints to a finite set. This set of approximating functions grows exponentially with the number of buffers and is larger than the piece-wise quadratic approximation that we propose. Our paper differs in that we use additional approximating functions, which are demonstrated numerically to give tighter bounds, and the development of constraint reduction methods for more general piece-wise quadratic and for certain exponential functions. Another difference is that they consider specific controls, rather than bounding average cost for a class of controls.

Average cost bounds have also been obtained for queueing networks using the achievable region method [4], [13] and Lyapunov functions [3]. These results are elegant but not very satisfactory from a numerical viewpoint. For example, the achievable region bound is quite loose in balanced heavy traffic, i.e., when more than one server has a traffic intensity near one. A duality relationship between the achievable region method and certain ALPs with quadratic approximation is noted in [12]. This relationship leads to the result that even the smallest, quadratic approximation ALP gives a tighter bound than the achievable region LP; see [20, Appendix A] and [24].

The rest of this paper is organized as follows. Section 2 defines the MQNET sequencing problem and the associated fluid control problem and Section 3 describes average cost ALPs. In Section 4 a variety of approximations and their constraint reductions are presented, including detailed analysis of a series queue and a reentrant line. Numerical results on the accuracy and speed of various ALPs are presented in Section 5. Some open questions are discussed in Section 6.

2 Open MQNET sequencing: Discrete and fluid models

In this section we describe the standard MQNET model and the fluid model associated with it. There are n job classes and m resources, or stations, each of which serves one or more classes. Associated with each class is a buffer in which jobs wait for processing. Let $x_i(t)$ be the number of class i jobs at time t , including any that are being processed. Class i jobs are served by station $\sigma(i)$. The topology of the network is described by the routing matrix $P = [p_{ij}]$, where p_{ij} is the probability that a job finishing service at class i will be routed to class j , independent of all other history, and the $m \times n$ constituency matrix with entries $C_{ji} = 1$ if station j serves class i and $C_{ji} = 0$ otherwise. If routing is deterministic, then $p_{i,s(i)} = 1$, where $s(i)$ is the successor of class i , and $p_{p(i),i} = 1$, where $p(i)$ as the predecessor of class i .

Exogenous arrivals occur at one or more classes according to independent Poisson processes with rate α_i in class i . Processing times are assumed to be independently exponentially distributed with mean $m_i = 1/\mu_i$ in class i . To create an open MQNET, the routing matrix P is assumed to be transient, i.e., $I + P + P^2 + \dots$ is convergent. As a result, there will be a unique solution to the traffic equation

$$\lambda = \alpha + P'\lambda$$

given by

$$\lambda = (I - P')^{-1}\alpha.$$

Here λ_i is the effective arrival rate to class i , including exogenous arrivals and routing from other classes, and vectors are formed in the usual way. The traffic intensity is given by

$$\rho = C \text{diag}(m_1, \dots, m_n)\lambda,$$

that is, ρ_j is the traffic intensity at station j . Stability requires that $\rho < 1$.

The network has sequencing control: each server must decide which job class to work on next, or possibly to idle. Preemption is allowed. Let $u_i(t) = 1$ if class i is served at time t and 0 otherwise. Admissible controls are nonanticipating and have

$$\begin{aligned} Cu(t) &\leq 1 \\ u_i(t) &\leq x_i(t). \end{aligned}$$

The first constraint states that a server's allocations cannot exceed one; the second prevents serving an empty buffer.

The objective is to minimize long-run average cost

$$J(x, u) = \limsup_{T \rightarrow \infty} \frac{1}{T} E_{x,u} \int_0^T c'x(t) dt.$$

Here $E_{x,u}$ denotes expectation given the initial state $x(0) = x$ and policy u . Consider only stationary Markov policies and write $u(t) = u(x(t))$. We use the uniformized, discrete-time Markov chain and assume that the

potential event rate is $\sum_{i=1}^n (\alpha_i + \mu_i) = 1$. Let $P_u = [p_u(x, y)]$ be the transition probability matrix under policy u and use the notation

$$(P_u h)(x) = \sum_y p_u(x, y) h(y).$$

Due to the linearity of P_u with respect to u , only extreme points $u_i = 0$ and $u_i = 1$ need be considered. Let $\mathcal{A}(x)$ be the set of feasible extreme point controls in state x and \mathcal{A} be their union.

Under the condition $\rho < 1$, the control problem has several desirable properties:

1. An optimal policy exists and its average cost is constant, $J_* = \min_u J(x, u)$ for all x .
2. There is a solution J_* and h_* to the average cost optimality equation

$$J + h(x) = c'x + \min_{u \in \mathcal{A}(x)} (P_u h)(x). \quad (1)$$

3. Under the additional condition that h is bounded below by a constant and above by a quadratic, there is a unique solution J_* and h_* to (1) satisfying $h_*(0) = 0$. Furthermore, J_* is the optimal average cost, any policy

$$u_*(x) = \arg \min_{u \in \mathcal{A}(x)} (P_u h_*)(x)$$

is optimal, and h_* is the differential cost of this policy,

$$h_*(x) = \limsup_{T \rightarrow \infty} \left(E_{x, u_*} \int_0^T c'x(t) dt - E_{0, u_*} \int_0^T c'x(t) dt \right). \quad (2)$$

Properties (1) and (2) can be established using general results for MDPs as in [27, Theorems 7.2.3 and 7.5.6]. For networks, properties (1) and (2) are shown in [15, Theorem 7]; (3) is obtained by applying standard verification theorems to networks; see, e.g., [14, Theorem 2.1 and Section 7] and [16, Theorem 10.7].

A natural starting point in approximating the differential cost function is the associated fluid model. In this model all transitions are replaced by their mean rates and a continuous state $q_i(t) \in \mathbf{R}_+$ is used. In a fluid control problem, for each initial state there is a time horizon T such that $q(t) = 0$ for all $t \geq T$. The fluid control problem corresponding to (1) is

$$\begin{aligned} \text{(FCP)} \quad V(x) &= \min \int_0^T c'q(t) dt \\ \dot{q}(t) &= Bu(t) + \alpha \\ Cu(t) &\leq 1 \\ q(0) &= x \\ q(t) &\geq 0, \quad u(t) \geq 0, \end{aligned}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)'$ and $B = (P' - I)\text{diag}(\mu_1, \dots, \mu_n)$. An optimal $u(t)$ can be chosen so that it is piecewise constant, making $q(t)$ piecewise linear with $\dot{q}(t)$ existing except on a set of zero measure. We will use the fluid ‘‘cost to drain’’ $V(x)$ to guide our approximation of $h(x)$. The motivation for this approximation is

[15, Theorem 7(iv)], based on [14, Theorem 5.2]. It establishes the following connection between the discrete and fluid cost functions:

$$\lim_{\theta \rightarrow \infty} \frac{h_*(\theta x)}{V(\theta x)} = 1, \quad (3)$$

i.e., the differential cost is dominated by the fluid cost as queue lengths increase; see [31] for discussion of the policy implications.

The fluid cost V is also piece-wise quadratic and \mathcal{C}^1 . The quadratic regions depend on the optimal fluid policy $u(x)$, which partitions \mathbf{R}_+^n into a finite number of control switching sets where the control is constant. Each switching set is a convex polyhedral cone emanating from the origin. In particular, if a switching set contains x , it also contains θx , $\theta > 0$. Switching sets can be subdivided according to the (finite) sequence of switching sets that a trajectory enters next. In a region, say S_k , where a certain sequence of switching sets will be visited, V is quadratic and we write

$$V(x) = \frac{1}{2}x'Q_kx, \quad x \in S_k. \quad (4)$$

Thus, (3) implies that

$$h_*(x) = \frac{1}{2}x'Q_kx + o(|x|^2), \quad x \in S_k. \quad (5)$$

Fundamentally, V is quadratic in these regions because $\dot{q}(t)$ is constant in a switching set, so that the integrand in (FCP) is piece-wise linear.

3 Approximate LP: Average cost bounds

In this section we describe a general method for constructing a linear program in a small number of variables that approximates the differential cost and places a lower bound on average cost. It is well-known that, for finite state spaces, an inequality relaxation of Bellman's equation gives an equivalent LP in the same variables,

$$\begin{aligned} \text{(LP)} \quad & \max J \\ \text{s.t.} \quad & J + h(x) \leq c'x + (P_u h)(x) \quad \text{for all } x \in \mathbf{Z}_+^n, u \in \mathcal{A}(x) \\ & h(0) = 0. \end{aligned}$$

An additional condition on h is needed because of the countable space: For some $L_1, L_2 > 0$,

$$-L_1 \leq h(x) \leq L_2(1 + |x|^2). \quad (6)$$

Appendix A shows that (1) is equivalent to (LP) and (6). This *exact LP* has one variable for every state. To create a tractable LP, the differential cost can be approximated by a linear form

$$h_*(x) \approx \sum_{k=1}^K r_k \phi_k(x) = (\Phi r)(x) \quad (7)$$

using some small set of basis functions ϕ_k and variables r_k . Assume that $\phi_k(0) = 0$. The resulting *approximate*

mate LP is

$$\begin{aligned}
(\text{ALP}) \quad \underline{J} &= \max J \\
\text{s.t.} \quad J + (\Phi r)(x) &\leq c'x + (P_u \Phi r)(x) \quad \text{for all } x \in \mathbf{Z}_+^n, u \in \mathcal{A}(x) \\
-L_1 &\leq (\Phi r)(x) \leq L_2(1 + |x|^2) \quad \text{for all } x \in \mathbf{Z}_+^n.
\end{aligned}$$

The bounds L_1 and L_2 may depend on r ; all that is needed is that the bound applies to each ϕ_k . Since (ALP) is equivalent to the exact LP with the constraints (7) added, the exact LP is a relaxation. Hence, (ALP) gives a lower bound, $J_* \geq \underline{J}$. (ALP) is feasible, so it has an optimal solution, say r_* .

To compute an upper bound, one must identify a policy or set of policies. Of course, a policy u can be simulated to estimate its average cost J_u . Alternatively, the following upper bound ALP can be used:

$$\begin{aligned}
(\text{ALP}_{UB}) \quad \bar{J} &= \min J \\
\text{s.t.} \quad J + (\Phi r)(x) &\geq c'x + (P_u \Phi r)(x) \quad \text{for all } x \in \mathbf{Z}_+^n \\
-L_1 &\leq (\Phi r)(x) \leq L_2(1 + |x|^2) \quad \text{for all } x \in \mathbf{Z}_+^n.
\end{aligned}$$

Then $\bar{J} \geq J_u \geq J_*$. Note that in addition to reversing the inequality in (ALP) and minimizing, (ALP_{UB}) considers a single policy. Including constraints for a class of policies, e.g., nonidling, is used to check stability.

Any differential cost approximation h defines an h -greedy policy

$$u_h(x) = \arg \min_{u \in \mathcal{A}(x)} (P_u h)(x).$$

The approximation architecture Φ restricts the greedy policy to a certain class of policies. For example, a quadratic approximation architecture implies linear boundaries between control regions. The greedy policy from (ALP) may have poor performance or not even be stabilizing. However, in some cases it has good performance, so we use it in (ALP_{UB}) to construct an upper bound on average cost.

The numerical tests in Section 5 use (ALP) and (ALP_{UB}). However, after solving (ALP) another approach to an upper bound is to use Bellman error, defined for any J and h as

$$\begin{aligned}
B(x) &= \min_{u \in \mathcal{A}(x)} (P_u h)(x) - h(x) + c'x - J \\
&= (P_{\tilde{u}} h)(x) - h(x) + c'x - J \\
&= [(P_{\tilde{u}} - I)h](x) + c'x - J,
\end{aligned}$$

where \tilde{u} is an h -greedy policy. If J, h satisfy the constraints (ALP) then $B(x) \geq 0$ is the minimum slack of constraints for that x . Under the assumption that the h -greedy policy is stabilizing, we can bound the average cost error $\underline{J} - J_*$ using Bellman error.

Proposition 1 *Let J, r be feasible for (ALP), $h = \Phi r$, and \tilde{u} be an h -greedy policy. Assume*

- A1. \tilde{u} is stabilizing. Let $E_{\tilde{u}}$ denote expectation with respect to a stationary distribution for policy \tilde{u} .
- A2. $J_{\tilde{u}} < \infty$ and $E_{\tilde{u}}(B) < \infty$.

Then

$$J_* - J = (J_* - J_{\tilde{u}}) + E_{\tilde{u}}(B) \geq E_{\tilde{u}}(B). \quad (8)$$

Proof. By the definition of B ,

$$J + h(x) = c'x - B(x) + \min_{u \in \mathcal{A}(x)} (P_u h)(x),$$

so J, h satisfy (1) for the perturbed problem with cost $c'x - B(x)$. Since (ALP) includes the growth condition (6), J is the average cost for this problem under policy \tilde{u} :

$$J = E_{\tilde{u}}[c'x - B(x)] = J_{\tilde{u}} - E_{\tilde{u}}(B)$$

which implies (8). ■

To compute the expectation in (8) one could simulate the policy \tilde{u} . The term in parentheses is the suboptimality of policy \tilde{u} . ALP policies can be unstable or have large suboptimalities, so the bound will not be useful in all examples. Other error bounds are proposed in [6] and [9] for variants of the ALP approach.

The quadratic bound on h (6) implies that, at least for sufficiently smooth h , Bellman error is bounded by a linear function. The precise form of B for quadratic h is shown in Section 4.1. A linear bound also holds for the other approximation architectures that we propose; the fundamental reason is that B depends on h only through the differences $h(x + e_i) - h(x)$, where e_i is a unit vector with i th component equal to 1. A linear bound on h has the advantage that the assumption $E_{\tilde{u}}(B) < \infty$ in Proposition 1 is no stronger than $J_{\tilde{u}} < \infty$.

If the approximation architecture happens to include the exact differential cost, (ALP) will find it.

Proposition 2 *If either $h_* = \Phi r$ for some r or (ALP) has a binding constraint for each state x then*

(i) $J_* = \underline{J}$ and $h_* = \Phi r_*$.

(ii) *If $\{\phi_k\}$ are linearly independent on \mathbf{Z}_+^n , then (ALP) has a unique optimal solution.*

Proof. Suppose $h_* = \Phi r$. Then J_* and Φr satisfy (1) and are optimal for (ALP). By assumption, $(\Phi r_*)(0) = 0$ and uniqueness of solutions to (1) implies that J_* and r are the unique optimal solution to (ALP), i.e., $r_* = r$ and (i) holds.

Now suppose (ALP) has a binding constraint for each state. Then \underline{J} and Φr_* satisfy (1) for each x , with the minimum achieved by the action $u(x)$ with the binding constraint in (ALP), again implying (i). (ii) follows easily from (i). ■

The approximate LP is still not a manageable size because it has one constraint for each state-action pair. In Section 4, various constraint sets are algebraically reduced or approximated by a smaller set of constraints.

4 Differential cost approximation and constraint reduction

This section considers several bases Φ to approximate the differential cost and demonstrates how the constraints of the resulting ALP can be algebraically reduced to a small, or at least more easily approximated,

set. In Section 4.1, constraint reduction is given for quadratic approximation. Section 4.2 introduces exponential approximating functions and a method of reducing the constraints, using a series queue as an example. A method of reducing piece-wise quadratic approximations, which are suggested by fluid models, is presented in Section 4.3. A larger class of indicator functions is proposed in Section 4.4.

4.1 Quadratic approximation

Consider the quadratic differential cost approximation

$$h(x) = \frac{1}{2}x'Qx + px \quad (9)$$

where $Q = [q_{ij}]$ is symmetric. This approximation is motivated by (5). It is also interesting to note that for a single uncontrolled queue

$$h_*(x) = \frac{1}{2(\mu - \alpha)}(x^2 + x). \quad (10)$$

The quadratic term in (10) matches the fluid value function; the effect of randomness is to shift the fluid value function $1/2$ unit to the left.

The constraints (ALP) can be reduced to a finite set for quadratic h [20, Appendix A]. To simplify notation, consider only deterministic routing. First, we write the constraints as

$$J \leq \sum_{i=1}^n (c_i x_i + \alpha_i [h(x + e_i) - h(x)] + u_i \mu_i [h(x - e_i + e_{s(i)}) - h(x)]) \quad \text{for all } x \in \mathbf{Z}_+^n, u \in \mathcal{A}(x). \quad (11)$$

Unlike a discounted model, only differences in h appear in these constraints, simplifying the analysis. It is convenient to let $x = z + u$, so that a control u is feasible for all $z \in \mathbf{Z}_+^n$. Substituting (9) into (11) yields

$$J \leq d^u + c^{u'} z \quad \text{for all } z \in \mathbf{Z}_+^n, u \in \mathcal{A} \quad (12)$$

where

$$\begin{aligned} c_i^u &= c_i + \sum_{j=1}^n [\alpha_j q_{ij} + u_j \mu_j (q_{i,s(j)} - q_{ij})] \\ d^u &= \sum_{i=1}^n [u_i (c_i^u + \mu_i (\frac{1}{2} q_{ii} + \frac{1}{2} q_{s(i),s(i)} - q_{i,s(i)} + p_{s(i)} - p_i)) + \alpha_i (\frac{1}{2} q_{ii} + p_i)] \end{aligned}$$

and $c^u = [c_i^u]$. For (12) to hold for all z , the right hand side must be nondecreasing in z_i . Hence, (12) is equivalent to

$$J \leq d^u \quad (13)$$

$$c_i^u \geq 0 \quad \text{for } i = 1, \dots, n, u \in \mathcal{A} \quad (14)$$

If the optimal policy is nonidling, then for a given control u , (14) is only needed for i in

$$N(u) = \left\{ i : \sum_{j:\sigma(i)=\sigma(j)} u_j = 1 \right\},$$

i.e., the classes served by busy stations. Under nonidling there are only $|N(u)| + 1$ constraints for each u instead of $n + 1$.

The number of constraints in (13) and (14) is typically exponential in n because of the number of actions $|\mathcal{A}|$. Nevertheless, the reduced ALP is small enough to solve for moderate n .

For quadratic h , Bellman error has the form

$$B(x) = \min_{u \in \mathcal{A}(x)} d^u + (c^u)'(x - u) - J,$$

which is the minimum of linear functions.

4.2 Exponential approximations and a series queue

Although quadratic approximation architectures are efficient, the error in average cost is often fairly large as shown in Section 5. In this section we introduce exponential approximating functions for a general network. A more detailed analysis, with constraint reduction, is given for a series queue.

Numerical experience suggests that a quadratic approximation misses important features of h_* in states where certain queue lengths are small or zero. Figure 1 shows the residual when a quadratic is fit to h_* over the region graphed for the series queue with $\mu_1 > \mu_2$ of Section 5. The residuals, plotted on the z axis, are small compared to h_* ; the largest value of h_* on this grid is over 500. The percent residual is larger when x is small and particularly when x_2 is small. The optimal stationary distribution also has larger probabilities in these states. In fact, above a switching curve where server 1 idles the probabilities are zero. In this example, the switching curve limits $x_2 \leq 5$ when $x_1 = 1$ and $x_2 \leq 9$ when $x_1 = 10$. Thus, additional functions to approximate the complex shape of h_* when x_2 is small appear important.

For general networks, to emphasize states with small x_i , we will use the function

$$\phi_i(x) = \beta_i^{x_i}, \tag{15}$$

where $\beta_i < 1$. There is a natural choice of β for some classes. If class i has a predecessor class, no arrivals, and $\mu_{p(i)} > \mu_i$ set $\beta = \mu_i/\mu_{p(i)}$. Then

$$\begin{aligned} [(P_u - I)\phi_i](x) &= \left[(\alpha_i + u_{p(i)}\mu_{p(i)})(\beta_i - 1) + u_i\mu_i(1/\beta_i - 1) \right] \phi_i(x) \\ &= \left[u_{p(i)}\mu_{p(i)}(\mu_i/\mu_{p(i)} - 1) + u_i\mu_i(\mu_{p(i)}/\mu_i - 1) \right] \phi_i(x) \\ &= (u_{p(i)} - u_i)(\mu_i - \mu_{p(i)})\phi_i(x). \end{aligned} \tag{16}$$

Note that if the two classes have different servers, $\sigma(p(i)) \neq \sigma(i)$, then in states where both are being served, $u_{p(i)} = u_i = 1$ and (16) is equal to 0, i.e., (16) is in the kernel of the generator P_u . This has two advantages. First, as shown below, it allows constraint reduction. Second, it can influence Bellman error in states where

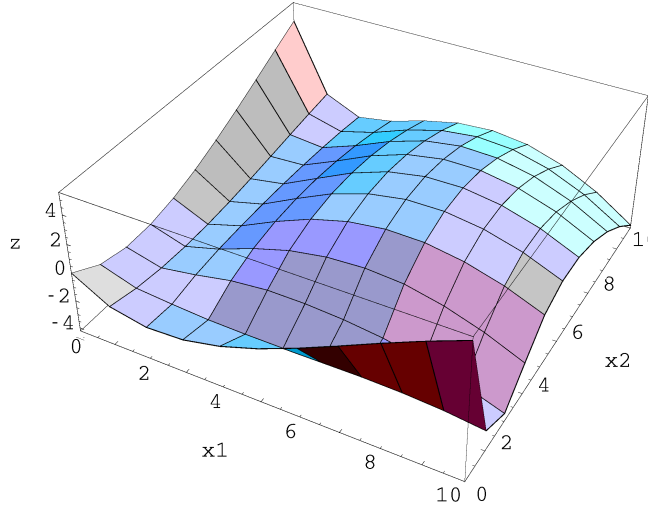


Figure 1: Residual (z) in the best quadratic fit to h_* for the series queue.

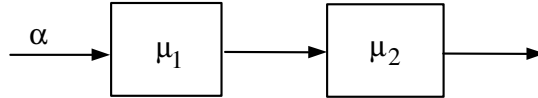


Figure 2: Two-stage series queue.

class i is not being served without affecting it in states where classes i and $p(i)$ are being served. For the switching curve policies typical of these networks, this would mean influencing Bellman error in states with $x_{p(i)}$ at or above the class i switching curve.

Motivated by the complex shape in Figure 1, we also use functions of the form

$$\phi_{ij}(x) = x_j \beta^{x_i}. \quad (17)$$

To illustrate the constraint reduction possible for such functions, consider a series queue with arrivals at rate α to the first queue (Figure 2). For this problem, (11) is

$$\begin{aligned} J \leq & c'x + \alpha [h(x + e_1) - h(x)] + u_1 \mu_1 [h(x - e_1 + e_2) - h(x)] \\ & + u_2 \mu_2 [h(x - e_2) - h(x)]. \end{aligned} \quad (18)$$

We consider the case $c_1 < c_2$, so that station 1 might idle (station 2 is always nonidling), and $\alpha < \mu_2 \leq \mu_1$, so that the fluid policy is greedy: idle 1 when $x_2 > 0$ [2]. Use the approximation

$$h(x) = \frac{1}{2}x'Qx + px + r_1\phi_2(x) + r_2\phi_{21}(x) + r_3\phi_{22}(x) \quad (19)$$

i.e., the functions (15) and (17) with $i = 2$ are added to the quadratic. Set $\beta \equiv \beta_2 = (\mu_2/\mu_1)$. Substituting (19) into (18) and again using $x = z + u$, (18) has the form

$$J \leq d^u + c^u z + (\zeta^u + \xi^u z)\beta^{z_2+u_2} \quad (20)$$

for all $z \in Z_+^2$ and all u that are nonidling at station 2. Here d^u , c^u , ζ^u and ξ^u are linear functions of the variables p , Q , and r ; see Appendix B. For $u = (1, 1)$, $\xi^{(1,1)} = 0$, so as $z_i \rightarrow \infty$ we must have

$$c_i^{(1,1)} \geq 0, \quad i = 1, 2. \quad (21)$$

Given (21), (20) is tightest at $z_1 = 0$ for each z_2 . However, depending on the value of $\zeta^{((1,1))}$ it could be tightest at any z_2 . Thus, we will approximate (ALP) by including (20) at $z_1 = 0$, $z_2 = 0, \dots, N-1$ for some N . Now consider $u = (0, 1)$. For each z_2 , the z_1 coefficient must be nonnegative,

$$c_1^{(0,1)} + \beta\xi_1^{(0,1)}\beta^{z_2} \geq 0.$$

Because of the monotonicity in z_2 , this is equivalent to

$$c_1^{(0,1)} + \beta\xi_1^{(0,1)} \geq 0 \quad (22)$$

and $c_1^{(0,1)} \geq 0$. Letting $z_2 \rightarrow \infty$ in (20) gives another constraint, so we have

$$c_i^{(0,1)} \geq 0, \quad i = 1, 2. \quad (23)$$

Given (22) and (23), (20) is tightest at $z_1 = 0$ but, depending on $\zeta^{(0,1)}$ and $\xi_2^{(0,1)}$, could be tightest at any z_2 , so we include (20) at $u = (0, 1)$, $z_1 = 0$, and $z_2 = 0, \dots, N-1$. Next, for $u = (1, 0)$ we must have $z_2 = 0$. For (20) to hold as $z_1 \rightarrow \infty$, we must have

$$c_1^{(1,0)} \geq 0. \quad (24)$$

In light of (24), (20) is tightest at $z_1 = 0$, so we include (20) at $u = (1, 0)$ and $z = (0, 0)$. Finally, we include (20) at $u = z = (0, 0)$.

To summarize, the approximate reduced ALP contains the $2N + 8$ constraints (20) at $u = (1, 1)$, $z_1 = 0$, $z_2 = 0, \dots, N-1$; $u = (0, 1)$, $z_1 = 0$, $z_2 = 0, \dots, N-1$; $u = (1, 0)$, $z = (0, 0)$; and $u = z = 0$; plus (21)-(24). Call this relaxation ALP(N). The following proposition shows that ALP(N) is exact for some N . In practice, a quite small N is usually sufficient.

Proposition 3 *Let \underline{J}_N be the optimal value of ALP(N) for the series queue. For some M , $\underline{J}_N = \underline{J}$ for all $N \geq M$.*

The proposition follows from the fact that the limiting constraints as $x_2 \rightarrow \infty$, namely, (21)-(24), are

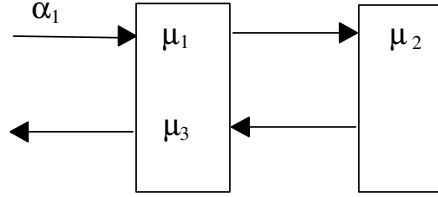


Figure 3: Three-class, two-station reentrant line.

incorporated in ALP(N).

For the series queue with the h approximation (19), we were able to reduce an infinite constraint set on \mathbf{Z}_+^2 to an infinite constraint set on \mathbf{Z}_+ for which a small number of constraints usually were equivalent. This partial reduction is also possible on larger problems. For the three-class reentrant line of Section 4.3, the analogous h approximation includes (15) and (17) for $i = 2, 3$. The constraint set on \mathbf{Z}_+^3 reduces to a constraint set on \mathbf{Z}_+^2 (\mathbf{Z}_+ if only the functions (15) are used).

Another advantage of (19) is that the greedy policies for this approximation include more realistic policies. When h is quadratic, the greedy policy has a linear switching curve; (19) allows more general shapes. For example, consider just ϕ_2 by setting $r_2 = r_3 = 0$. Server 1 is busy in the greedy policy when

$$h(x - e_1 + e_2) - h(x) = (-q_{11} + q_{12})x_1 + (q_{22} - q_{12})x_2 + \frac{1}{2}q_{11} + \frac{1}{2}q_{22} - q_{12} - p_1 + p_2 - r_1(1 - \beta)\beta^{x_2} < 0. \quad (25)$$

Solving ALP(N) numerically often leads to $q_{22} = q_{12}$ and $r_1 < 0$; then (25) reduces to

$$x_2 < \ln(x_1 + A) / \ln \beta + B$$

for some A and B . Numerical experience and [18] suggest a logarithmic form to the optimal switching curve. Thus, the approximation architecture has the potential to produce realistic switching curves.

4.3 Piece-wise quadratic approximations and a reentrant line

Piece-wise quadratic approximations are motivated by the form of $V(x)$ in the fluid model. The fluid model of the series queue in Section 4.2 with $\mu_1 \geq \mu_2$ has a greedy optimal policy and quadratic V . In contrast, if $\mu_1 < \mu_2$, the fluid policy has a linear switching curve through the origin and V is piece-wise quadratic with two quadratic regions. There is a solidarity between the latter type of fluid policy and the optimal policy: in this case the fluid policy gives richer information about the optimal policy [31] and tends to perform better [30]. We formulate the ALP using the piece-wise quadratic regions from the fluid and propose an approximate constraint reduction. The method is applied to the three-class reentrant line studied by Weiss [32].

Quadratic region ($x > 0$)	Control region visited next		
	State	Station 1 serves	\dot{q}
$S_1 : x_3 > \gamma x_1 + \frac{\alpha\gamma + \mu_3 - \mu_2}{\mu_2} x_2$	$x_2 = 0, x_3 > \gamma x_1$	3	$(\alpha, 0, \mu_2 - \mu_3)$
$S_2 : x_3 \leq \gamma x_1 + \frac{\alpha\gamma + \mu_3 - \mu_2}{\mu_2} x_2$ and $x_3 > \frac{\mu_3 - \mu_2}{\mu_2} x_2$	$x_2 = 0, x_3 \leq \gamma x_1$	1 and 3	$(\alpha - \mu_2, 0, \mu_2 - \mu_3(1 - \frac{\mu_2}{\mu_1}))$
$S_3 : x_3 \leq \frac{\mu_3 - \mu_2}{\mu_2} x_2$	$x_2 > 0, x_3 = 0$	3 and idle	$(\alpha, -\mu_2, 0)$

Table 1: Quadratic regions of the fluid cost in the reentrant line example

Figure 3 shows a three-class, two-station line where jobs arrive at rate α to class 1. Station 2 serves only class 2 and is the bottleneck, $m_2 > m_1 + m_3$, where $m_i = 1/\mu_i$ is the mean service time for class i . Costs are constant, $c_i = 1$, so the only decision is whether to serve class 1 or 3 at station 1. As Weiss shows, when $x_2 = 0$ the fluid policy makes a trade-off between serving class 3, which starves the bottleneck, and serving class 1, feeding the bottleneck. Class 3 is given priority when $x_3 \leq \gamma x_1$, where

$$\gamma = \frac{1}{1 - \alpha/\mu_2} \left(\frac{m_2 - m_1 - m_3}{m_1 + m_3} \right).$$

When $x > 0$ class 3 is served.

Although the control is constant on $x > 0$, V divides this region into three *quadratic* regions according to which of three controls will be used next on a trajectory starting from x . The correspondence between quadratic regions and control regions is shown in Table 1. One can verify that these are the three quadratic regions by following the trajectories. Trajectories in the first control region in Table 1 enter the second control region next; the second and third region feed into the control region $x_2 = 0$ and $x_3 = 0$, which leads to $x = 0$. Note that this optimal policy idles machine 1 in the third control region, but this is just for convenience; a nonidling optimal policy also exists. Since V is continuous (in fact it is C^1), the regions S_k can be extended to include $x_2 = 0$, fully defining V . Hence, we approximate h as quadratic on each of these regions:

$$h(x) = \frac{1}{2} x' Q^k x + p^{k'} x + f^k, \quad x \in S_k, \quad (26)$$

where S_k , $k = 1, \dots, 3$ are defined in Table 1. Note that we do not require h to be continuous, so the treatment of boundaries matters. The assignment of boundaries in Table 1 was chosen for consistency with the control regions.

Now consider constraint reduction for a general network using (26). Assume that there are finitely many S_k , $k = 1, \dots, \kappa$, each a polyhedral cone from the origin of full dimension. A dual method similar to [21, Section 3.4] will be used to reduce the constraints to a finite set; however, certain approximations are needed to make the constraints tractable. For each control, this approach defines sets of states in which the form of the constraints (11) is constant. Given the control, the transition probabilities are translation invariant. Let $\{\delta_l\}$, $l = 1, \dots, q$ be the transitions, i.e., $p_u(x, x + \delta_l) > 0$ for some u and all x such that $u \in \mathcal{A}(x)$. Let $\psi = (u, k_0, k_1, \dots, k_q)$ and define

$$X^\psi = \{x \in S_{k_0} \cap Z_+^n : x + \delta_l \in S_{k_l} \text{ for all } l \text{ such that } p_u(x, x + \delta_l) > 0\}.$$

There is one index ψ for each combination of control u , quadratic region S_{k_0} of the current state, and

quadratic region S_{k_l} of possible next states. If transition δ_l does not occur under u , then $k_l = k_0$.

To illustrate these definitions in the reentrant line example, number the service transitions $l = 1, 2, 3$ and the arrival transition $l = 4$. Consider, for example, $u = (0, 1, 1)$ and $k_0 = 3$, i.e., $x \in S_3$ (see Table 1). Then $k_1 = 3$ because class 1 is not served and $k_3 = k_4 = 3$ because these transitions cannot leave S_3 . However, a class 2 service completion could stay in S_3 ($k_2 = 3$), enter S_2 ($k_2 = 2$), or, for certain parameter values, enter S_1 ($k_2 = 1$). Specifically, if $\mu_3 \geq 2\mu_2$ and $x = (0, 1, 1)$ then $x \in S_3$ but $x + \delta_2 = (0, 0, 2) \in S_1$, i.e., $x \in X^\psi$ where $\psi = (u, 3, 3, 1, 3, 3)$. Because S_1 and S_3 only meet at the origin, X^ψ can contain only states near the origin. In general, if ψ contains $k_l \neq k_0$ then X^ψ lies within one transition of the hyperplane separating S_{k_0} and S_{k_l} .

Again using $x = z + u$, let $Z^\psi = \{z : z + u \in X^\psi\}$. The constraints have the form

$$J \leq d^\psi + c^{\psi'} z + \frac{1}{2} z' M^\psi z, \quad z \in Z^\psi \quad (27)$$

where d^ψ , $c^{\psi'}$, and M^ψ are linear functions of Q^k , p^k , and f^k . The quadratic term M^ψ is symmetric. It appears because of transitions between regions S_k . The first approximation is to remove the integer restriction by allowing $z \in \bar{Z}^\psi$, where \bar{Z}^ψ is a polyhedron, say $\{z \in R^n : A^\psi z \geq b^\psi, z \geq 0\}$, whose lattice points are (nearly) the set Z^ψ . For simplicity, we allow lattice points on the boundary of \bar{Z}^ψ that are not in Z^ψ . This overlap could be avoided by adding more cutting planes. Also, because S_k has full dimension and the control u is feasible at all $z \geq 0$, there is no need for equality constraints in \bar{Z}^ψ . If nonidling controls are desired, constraints of the form $z_i = 0$ can be enforced by removing these variables.

Checking (27) exactly for a given d^ψ , $c^{\psi'}$, and M^ψ is related to determining if M^ψ is copositive; instead, following [21], we impose the stronger, simpler conditions

$$M^\psi \geq 0 \quad (28)$$

and

$$\begin{aligned} J &\leq d^\psi + c^{\psi'} z \\ A^\psi z &\geq b^\psi \\ z &\geq 0. \end{aligned} \quad (29)$$

The key observation is that these constraints are colinear in z and the ALP variables. A dual can be constructed that separates z . Fixed values of J , Q^k , p^k , and f^k satisfy (29) if and only if, for each ψ , the LP

$$\begin{aligned} \min \quad & c^{\psi'} z \\ \text{s.t.} \quad & A^\psi z \geq b^\psi \\ & z \geq 0 \end{aligned}$$

has optimal value $w^\psi \geq J - d^\psi$, or equivalently, so does its dual

$$\begin{aligned}
\max \quad & b^{\psi'} y^\psi \\
\text{s.t.} \quad & A^{\psi'} y^\psi \leq c^\psi \\
& y^\psi \geq 0.
\end{aligned} \tag{30}$$

Thus, (30) and $w^\psi \geq J - d^\psi$ for all ψ are equivalent to (29). Reintroducing J , Q^k , p^k , and f^k as variables, the dual form of (ALP) is

$$\begin{aligned}
(\text{ALPD}) \quad \max \quad & J \\
\text{s.t.} \quad & A^{\psi'} y^\psi \leq c^\psi \\
& b^{\psi'} y^\psi \geq J - d^\psi \\
& M^\psi \geq 0 \\
& y^\psi \geq 0.
\end{aligned}$$

The two approximations made were restrictions of (ALP); hence, the optimal value \underline{J}^D of (ALPD) is also a lower bound, $\underline{J}^D \leq J_*$.

(ALPD) has roughly $n^2/2$ constraints for each ψ . It contains the roughly $\kappa n^2/2$ variables in (26) (recall that κ is the number of quadratic regions) plus one dual variable for each constraint used to define the polyhedra \overline{Z}^ψ . Both κ and the number of dual variables are generally exponential in n , but the latter grows much more quickly due to the large number of regions indexed by ψ . Hence, (ALPD) has many more variables than (ALP). We propose two reductions. First, if nonidling is assumed, then $z_i = 0$ for $i \notin N(u)$, and these z_i can be eliminated before forming the dual.

Second, (29) can be interpreted geometrically as checking $J \leq d^\psi + c^{\psi'} z$ for every extreme point z and $c^{\psi'} \beta \geq 0$ for every extreme direction β of \overline{Z}^ψ . Because the hyperplanes bounding each S_k pass through the origin, the ones bounding \overline{Z}^ψ pass within roughly one transition of the origin (there are points in Z^ψ within one transition of the S_k boundary). Thus, in a certain sense, the extreme points of \overline{Z}^ψ lie near the origin. Also, finding the extreme directions is made easier by the fact that the extreme directions of \overline{Z}^ψ are a subset of the extreme directions of S_{k_0} . In particular, \overline{Z}^ψ has the ones contained in the common boundary of S_{k_0} and all S_{k_l} (because there are transitions into S_{k_l} from \overline{Z}^ψ). The extreme directions for the example in this section are listed in Table 2. Checking extreme directions in the linear constraints (29) is an exact method; however, we will apply it as an approximate check of (27). Requiring (27) at $z = t\beta$ for all $t \geq 0$ results in quadratic constraints. For simplicity, we use the stronger conditions $c^{\psi'} \beta \geq 0$ and $\beta' M^\psi \beta \geq 0$.

These observations suggest the following approximation to (27). Find the extreme directions $\{\beta^{\psi,l}\}$ of \overline{Z}^ψ . The relaxation ALP(N) contains the constraints

$$\begin{aligned}
(27) \text{ for } z \in Z^\psi \text{ and } z_i \leq N - 1 \\
c^{\psi'} \beta^{\psi,l} \geq 0
\end{aligned} \tag{31}$$

$$(\beta^{\psi,l})' M^\psi \beta^{\psi,l} \geq 0 \tag{32}$$

Region	Extreme directions
S_1	$(0, 0, 1), (0, \mu_2, \alpha\gamma + \mu_3 - \mu_2), (1, 0, \gamma)$
S_2	$(1, 0, 0), (0, \mu_2, \alpha\gamma + \mu_3 - \mu_2), (1, 0, \gamma), (0, \mu_2, \mu_3 - \mu_2)$
S_3	$(1, 0, 0), (0, 1, 0), (0, \mu_2, \alpha\gamma + \mu_3 - \mu_2)$

Table 2: Edges of the quadratic regions in the reentrant line example

for all ψ and all directions $\beta^{\psi,l}$. The constraints (27) address the extreme points, while the limiting constraints (31) and (32) in the extreme directions allow faster convergence over N . ALP(N) has two limiting constraints for every extreme direction $\beta^{\psi,l}$, which is generally exponential in n , plus N^n constraints (27). However, it avoids the dual variables y^ψ , making it potentially more tractable than (ALPD).

Notice that ALP(N) is based on the exact constraints, not the linearization (29), suggesting that ALP(N) might give a tighter bound than (ALPD). However, because of the approximate treatment of limiting constraints ALP(N) may not converge to (ALP).

In this example, the fluid policy and quadratic regions were determined analytically. For larger problems, finding the fluid policy becomes intractable. An alternative is to analyze the two-station fluid workload relaxation in [17], for which an optimal policy can easily be found. A “greedy, workload constrained” translation of this policy from the workload space to the original state space is also given in [17]. It appears that the quadratic regions for this policy could be determined algorithmically by working backward from the origin and determining all sequences of control regions that can be visited, though we have not done so.

4.4 Indicator functions and state aggregation

Another strategy for refining the h approximation in specific states is to use functions that are only nonzero in one state or a small set of states. We test two methods, one using functions on individual states and one using rectangular “panels” of states. Several considerations guide the choice of states to focus on. First, states with larger probability in the stationary distribution under the optimal policy (or the h -greedy policy—see Proposition 1) are likely to affect \underline{J} more. Without knowing the particular network, we use the general principle that states with smaller queue lengths have higher probability; in particular, the high-probability states generally have at least one small queue length. Second, states whose constraints are very different than other constraints have more potential to affect \underline{J} . For a fixed action u , the importance of constraints with the smallest x , i.e., $x = u$, was seen in the constraint reduction of Sections 4.1 and 4.2. Hence, states with $x_i = 0$ or 1 for all i are likely to affect \underline{J} . Constraints with large total queue length are not as directly coupled to constraints for these states and, at least for the quadratic and exponential functions of Sections 4.1 and 4.2, vary less with x .

Based on these observations, the first set of approximating functions is

$$\phi_y(x) = 1_{\{x=y\}} \text{ for all } y \in Z_+^n \text{ such that } \sum_{i=1}^n y_i \leq M \quad (33)$$

for some M . For notational convenience, we have indexed the functions by the n -vector y . Using these indicator functions on individual states with total queue length up to M is equivalent to solving (1) exactly on these states; there is no Bellman error in these states. This set contains roughly M^n/n functions.

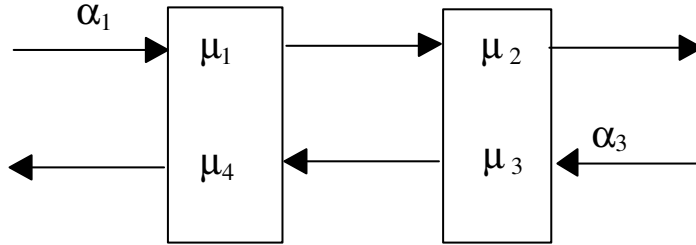


Figure 4: The Rybko-Stolyar network.

The second method, called paneling, aggregates states into sets that grow exponentially as the total queue length grows and uses a piece-wise constant function on these sets:

$$\phi_y^p(x) = 1 \text{ if } 2^{y_i-1} - \frac{1}{2} \leq x_i < 2^{y_i}, i = 1, \dots, n \text{ for all } y \text{ such that } 0 \leq y_i \leq M_i \quad (34)$$

for some $\{M_i\}$. Here ϕ_y^p is an indicator function on a rectangular panel and is equal to zero in other states. The panels cover the states with $x_i < 2^{M_i}$, dividing each x_i into the sets $\{0\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7\}$, etc. The assumption that ϕ_y^p is constant on a panel is arbitrary; other functions could be used.

5 Numerical Results

The tightness of the various ALP bounds was tested by computing the optimal average cost using DP value iteration on a truncated state space. The two networks of Sections 4.2 and 4.3 and the following four networks were included. In the arrival routing problem from [10], arrivals at rate α must be immediately routed to one of two classes, each with its own server. For an exponential differential cost approximation,

$$h(x) = \frac{1}{2}x'Qx + px + r_1x_2 \left(\frac{\mu_1}{\alpha}\right)^{x_1} + r_2x_1 \left(\frac{\mu_2}{\alpha}\right)^{x_2} + r_3 \left(\frac{\mu_1}{\alpha}\right)^{x_1} + r_4 \left(\frac{\mu_2}{\alpha}\right)^{x_2} \quad (35)$$

is used, analogous to (19). In the parallel server "N" network from [11], there are two classes and two servers. Server 1 can only serve class 1. Server 2 can serve class 1 or class 2. The previous two examples do not fit the formulation of Section 2, but the required modifications are easily made. The Rybko-Stolyar network, shown in Figure 4, is considered a challenging example because some static priority policies are not stabilizing. The six-class, two-station network of Figure 5 is the largest for which DP results are available; they are taken from [23]. We also report the size of the ALP and the solution time for larger series lines.

The parameters and values of the exponential base β_i for the examples are shown in Table 3. Note that the rates have not yet been scaled. For example, the arrival routing α and μ_i must be divided by their sum of 2.3. The value $\beta_i = \mu_i/\mu_{p(i)}$ from Section 4.2 is generally used when it applies; however, a maximum of 0.9 is applied, since we must have $\beta_i < 1$. For arrival routing, the exponential base is μ_i/α as in (35). For Rybko-Stolyar and the 6-class network, β_i was changed to 0.5, which gives a slightly better bound than the rule just described. For the "N" network a search was performed for the best value of β_i , improving the bound significantly. It should be noted that the parameter values for the Rybko-Stolyar and 6-class

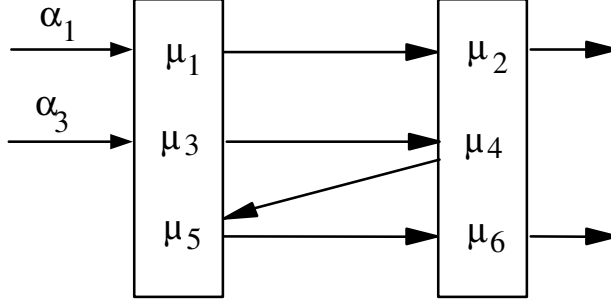


Figure 5: A six-class network.

	α	μ	c	ρ_{\max}	β
Series queue ($\mu_1 > \mu_2$)	1	1.5, 1.25	1, 2	0.8	.9, .83
Series queue ($\mu_1 < \mu_2$)	1	1.15, 1.4	1, 3	0.87	.9, .9
Series queue ($\mu_1 = \mu_2$)	varies	1, 1	1, 2	—	not used
2-class "N"	1.2, 0.4	1, 1, 1	1, 1, 1	0.8	.1, .1
Arrival routing	1	0.65, 0.65	1, 2	0.77	.65, .65
3-class reentrant line	0.1429	0.6, 0.16, 0.25	1, 1, 1	0.89	not used
Rybko-Stolyar	.0672, .0672	0.12, 0.12, 0.28, 0.28	1, ..., 1	0.8	.5, ..., .5
6-class network	6/140, 6/140	1/4, 1, 1/8, 1/6, 1/2, 1/7	1, ..., 1	0.6	.5, ..., .5

Table 3: Parameters of the examples

networks are taken from the literature (the arrival rates for Rybko-Stolyar were scaled to give $\rho_{\max} = 0.8$), the series queue ($\mu_1 > \mu_2$ and $\mu_1 = \mu_2$) and arrival routing parameters were set as typical values, but the series queue ($\mu_1 < \mu_2$), "N" system, and reentrant line parameters were selected after testing a variety of parameter values. The percent error for the other parameter values varied and was usually slightly larger than the reported cases.

The ALP with quadratic approximation (Section 4.1) is the smallest and has been solved on series lines of up to 17 classes; see Figures 6 and 7. For comparison, (ALP_{UB}) with quadratic approximation was also solved, using the policy found by (ALP) . Because h is quadratic, the ALP policy has polyhedral control regions (i.e., linear switching surfaces). We used the constraint reduction method developed in [21] for polyhedral control regions. The parameters used were $\alpha = 0.9$ for (ALP) and $\alpha = 0.5$ for (ALP_{UB}) , $\mu = (1 + 0.1(n - 1), \dots, 1)$ and $c = (2, \dots, 1)$, with the μ_i and c_i equally spaced. Run time is using CPLEX Version 9.0 on a 64-bit dual core processor and includes generating and solving the LP. The number of variables for the lower bound ALP is roughly $n^2/2$, while the upper bound ALP contains more variables than constraints (variables are added by the constraint reduction method). As a result, the upper bound ALP runs significantly slower. The lower bound ALP requires much more memory than time. In [29], larger networks are solved using column generation to reduce the memory requirement.

Next the effect of traffic intensity on the quadratic and exponential approximation (19) was tested. In Figure 8, $\rho = \alpha/\mu_2$ was varied while keeping μ_i fixed in the series queue with $\mu_1 > \mu_2$. The data below the line at 1.0 shows the accuracy of the lower bound, i.e., \underline{J}/J_* . The data above 1.0 shows the performance of the h -greedy policy for the ALP solution h , found using value iteration for this fixed policy. The percent

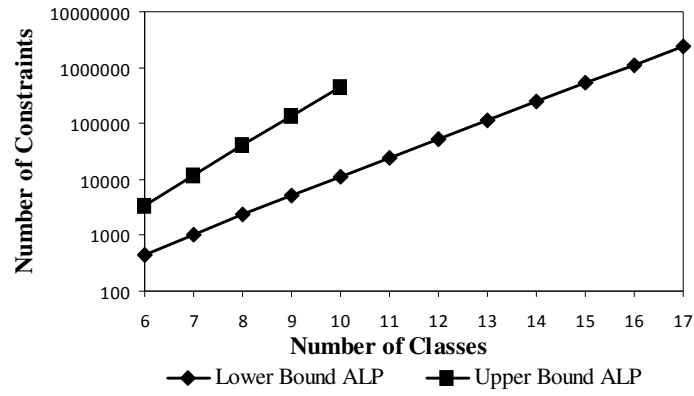


Figure 6: Size of the ALPs with quadratic approximation, series line.

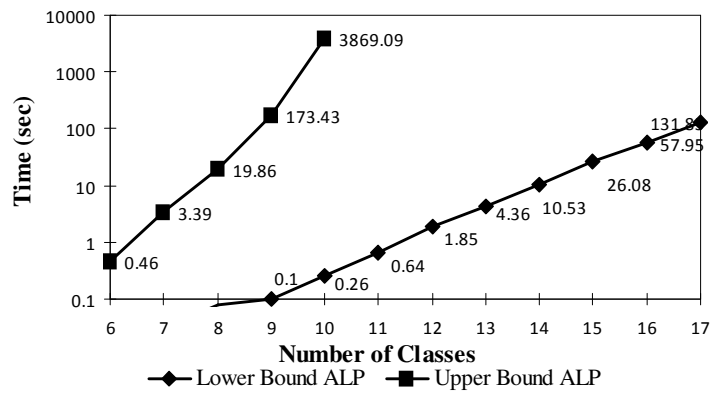


Figure 7: Run time for the ALPs with quadratic approximation, series line.

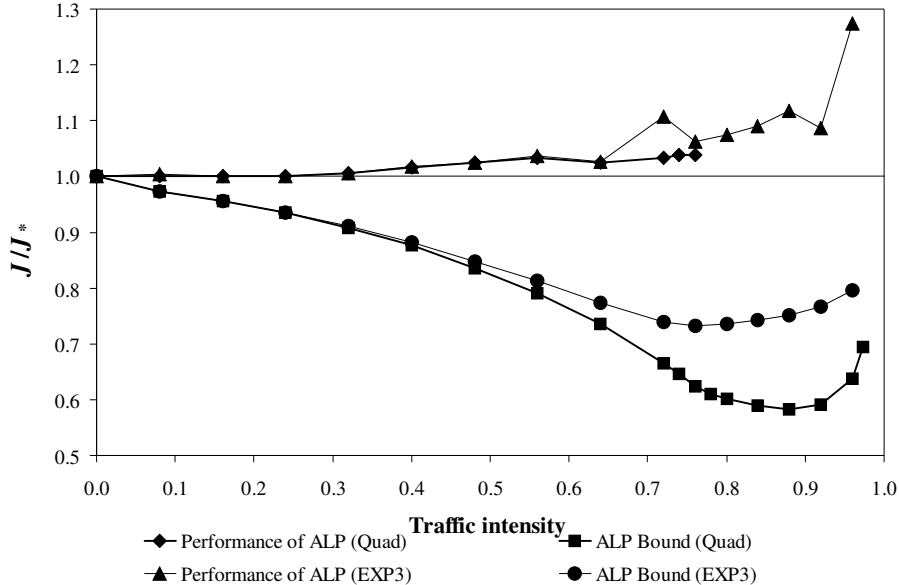


Figure 8: Performance of the ALP policy, two-station series line, $\mu_1 > \mu_2$.

error vanishes in light traffic. This is not surprising, since the five variables in (9) can fit h_* exactly in the six states with $x_1 + x_2 \leq 2$, which are a sufficient state truncation in light traffic. Although the DP can only be solved up to a certain ρ , the data suggests that percent error also vanishes in heavy traffic. This also is not surprising, since there is a single bottleneck at station 2. It is shown in [29] that the quadratic ALP gives a tighter bound than the achievable region method, which is known to have a vanishing percent error in heavy traffic [4]. Performance of the ALP policy is good, within 10% of optimal in most cases, except in heavy traffic. Performance of the quadratic approximation is not shown at traffic intensity above 0.75 because its policy is unstable. Surprisingly, the exponential approximation (19) sometimes has poorer performance than the quadratic, even though (19) includes quadratic functions. The quadratic ALP policy for this example has a simple form: server 1 is busy when $x_2 \leq 1$ and $x_2 > 0$, and this policy happens to perform fairly well. The series line with $\mu_1 < \mu_2$, though not shown, also has a single bottleneck and gives similar results except that performance of both ALP policies is better, with error under 5%, even in heavy traffic. In this example the optimal switching curve slopes up more steeply and is more closely matched by the quadratic ALP policy. Traffic intensity affects error in a similar way when the piecewise quadratic approximation (26) is used for the reentrant line, again with a single bottleneck station. Different behavior is seen for the series line with $\mu_1 = \mu_2$ (Figure 9). It has two bottlenecks. As traffic intensity increases, the accuracy of the ALP bound continues to degrade.

The size and accuracy of (ALP) for the suite of examples is shown in Table 4. Size is reported as rows (the number of constraints) by columns (the number of variables, which is the number of approximating functions plus one). The approximating functions used in addition to quadratic are the exponential functions (15) and (17), the piece-wise quadratic functions described in Section 4.3, or both. Two exceptions are the series

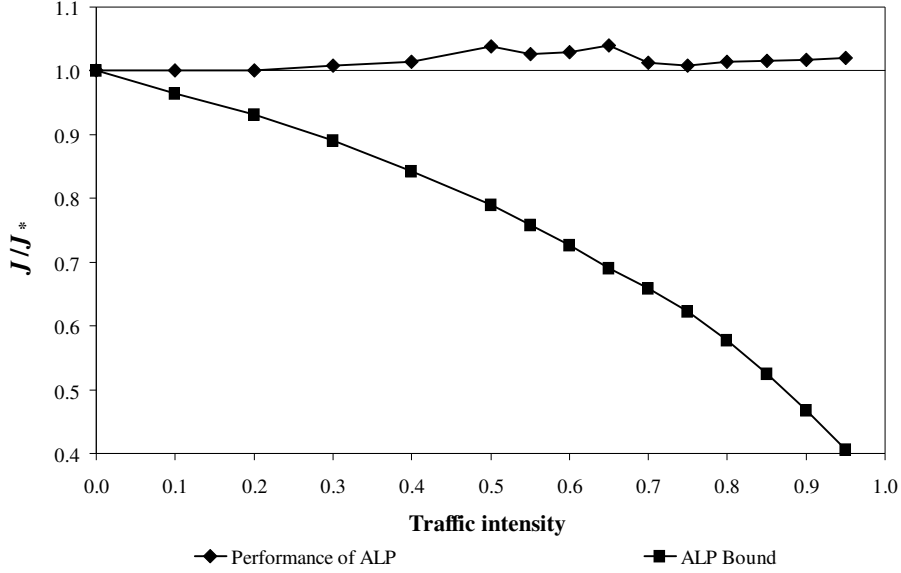


Figure 9: Performance of the quadratic ALP policy, two-station series line, $\mu_1 = \mu_2$.

queue, which uses (19), and arrival routing, which uses (35). The optimal average cost for the 6-class network is taken from [23].

The truncation method of Section 4.2 was used for the series queue ($\mu_1 > \mu_2$) and arrival routing. The number of constraints reflects the smallest N for which the solution to ALP(N) has no truncation error. For example, for the series queue $N = 3$ was used because all larger N give the same result. The large number of constraints for the piece-wise quadratic functions is because truncation was used, not the method of Section 4.3. For the Rybko-Stolyar and 6-class networks, constraint sampling was used (see [8]), introducing an additional source of error. Tests with different numbers of constraints suggest that the resulting \underline{J} is within 2% of the value for the ALP with all constraints.

Although the bounds on average cost are fairly loose, the LPs to obtain them are quite tractable due to the small number of variables. Particularly encouraging is the fact that accuracy improves rapidly when

	Optimal average cost	Error in ALP \underline{J} (size of LP)		Type of add'l function
		Quadratic	Quadratic + Add'l	
Series queue ($\mu_1 > \mu_2$)	9.31	-40% (13×6)	-27% (15×9)	EXP (19)
Series queue ($\mu_1 < \mu_2$)	13.50	-27% (13×6)	-18% ($321, 202 \times 14$)	both (19)
2-class "N"	4.54	-12% (11×6)	-1.8% ($15, 250 \times 12$)	EXP
Arrival routing	5.54	-19% (13×6)	-14% (75×10)	EXP (35)
3-class reentrant line	11.32	-19% (18×10)	-14% ($134, 553 \times 28$)	P-WQ
Rybko-Stolyar	6.87	-32% (45×15)	-21% ($500, 000 \times 35$)	EXP
6-class network	2.56	-19% (89×28)	-12% ($1, 000, 000 \times 70$)	EXP

Table 4: Accuracy of the ALP average cost

more functions are added. Adding three exponential functions to the quadratic ALP for the series queue with $\mu_1 > \mu_2$ cut the error from 40% to 27%. Adding three more exponential functions (15) and (17) reduced the error to 18%. For the series queue with $\mu_1 > \mu_2$, these errors are 27%, 18%, and 12%. The piece-wise quadratic functions were also fairly effective when added to the quadratic, reducing error from 19% to 14% for the reentrant line. For the series queue with $\mu_1 < \mu_2$, the piece-wise quadratic approximation (not shown) added 5 functions and reduced error from 27% to 21%. Based on Figure 8, these bounds should be tighter in light traffic or single-bottleneck heavy traffic.

The indicator functions described in Section 4.4 were also tested. Error in the average cost bound using indicator functions in individual states (33) or on panels (34), plus the quadratic and exponential terms (19), is shown in Figure 10 for the series line with $\mu_1 > \mu_2$. The number of variables in the ALP grows as indicator functions are added for states with larger total queue length. For comparison, the accuracy of using dynamic programming on a truncated state space, $x_i \leq N$ for some maximum buffer size N , is shown (in the DP there is one variable $h(x)$ for each state). The ALP achieves the same accuracy with an order of magnitude fewer variables. An error of less than 1% was achieved using the 104 indicator functions with $|x| < 13$ (a total of 113 basis functions). To achieve 1% accuracy using DP, a state space of $28^2 = 784$ states is required. The comparison in terms of solution time is similar when value iteration is used to solve the DP. Figure 11 makes the same comparison for the 3-class reentrant line. For this example the ALP does not include exponential functions and the parameter values in [5] were used, which differ from Table 3 in that $\mu_1 = \mu_3 = 22/63$ and $\mu_2 = 10/63$, giving $\rho_{\max} = 0.9$. Accuracy of 5% is achieved using 2,309 functions in the ALP and 32,768 variables in the DP. The larger number of variables needed reflects the higher traffic intensity and moving from two to three classes. The panel method only gives modest improvements in accuracy. We also tested piece-wise functions on panels of the form $(\sum_{i=1}^n x_i)\phi_y^p(x)$, $\max\{x_i\}\phi_y^p(x)$, and, for the series line, $V(x)\phi_y^p(x)$, where $V(x)$ is the fluid cost. Results depended very little the form of function; we report the best result for each number of functions. Adding more panels at larger x_i gives little improvement in accuracy, which is why panel results are not shown when the number of variables is large. We expect that better state aggregation methods could be found and could be useful for achieving moderate accuracy.

A quadratic upper bound ALP was also tested on these examples, using the policy from the quadratic lower bound ALP as described above. For the series line with $\mu_1 < \mu_2$, average cost was 33% above optimal. For the other examples, the upper bound ALP was infeasible, suggesting that the policy was unstable. Further testing has shown that at moderate or low traffic intensity the upper bound ALP is usually feasible (the upper bound results in Figure 7 are for $\rho_{\max} = 0.5$), though the bound has more error than the quadratic lower bound ALP.

6 Conclusion

We have demonstrated the feasibility of using ALP to compute lower bounds on optimal average cost for small to moderate size networks. The method requires only the solution of an LP. A policy can also be obtained which is useful for some networks. The ALPs have the following features:

- For all of the approximation architectures used, the bounds are tighter than the LP-based bounds in [4], [13].

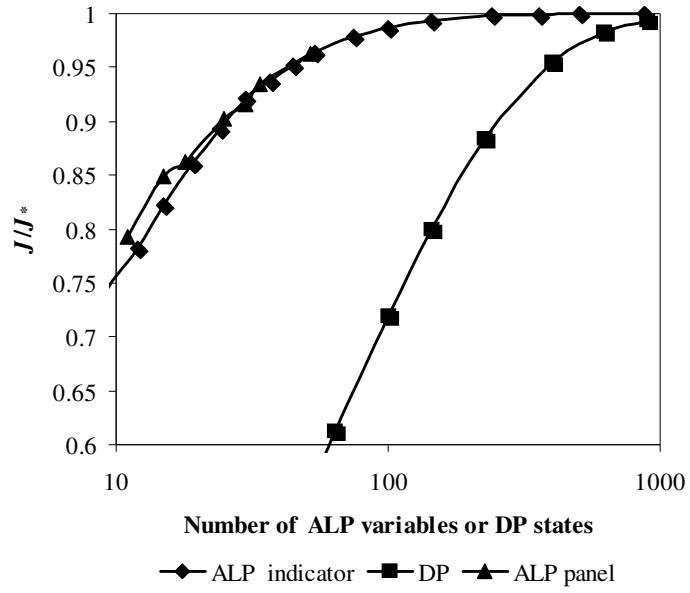


Figure 10: Accuracy of ALP with indicator functions and DP, series line, $\mu_1 > \mu_2$.

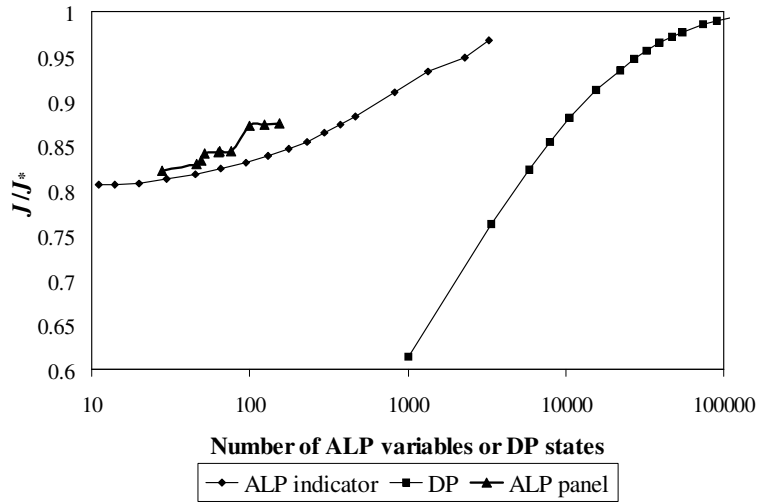


Figure 11: Accuracy of ALP with indicator functions and DP, reentrant line.

- The simplest, quadratic ALP has moderate accuracy and is fairly tractable. After constraint reduction, the number of constraints in the LP grows linearly the number of control actions and the number of buffers. Typically the number of actions is exponential in the number of buffers, but with a small base, and the resulting LP can be solved for much larger networks than can be solved exactly by DP.
- Accuracy can be improved significantly by adding a set of roughly n^2 exponential approximating functions (where n is the number of buffers). New constraint reduction techniques were developed that make ALPs with some of these exponential terms more tractable.
- Accuracy can also be improved by adding piece-wise quadratic approximating functions based on the associated fluid model. The number of approximating functions depends on the complexity of the fluid policy. General constraint reduction techniques were developed to help make these ALPs solvable.
- A sequence of ALPs with additional approximating functions that are indicators for individual states or sets of states up to a maximum buffer size can be used to obtain more accurate bounds. On small examples, the bounds require much less computation than using DP value iteration on a truncated state space; however, this advantage diminishes when more accuracy is desired. This set of functions is exponential in the number of buffers; the aggregation method reduces the base of the exponential growth, making the method practical for somewhat larger networks.

In addition, two upper bounds on optimal average cost were proposed. The first, which involves solving a second LP, gives fairly loose bounds, suggesting that simulation might be a better approach.

Several questions about the ALP approach remain open:

- More work is needed to determine larger sets of approximating functions that give tighter bounds. The quadratic, exponential, and piece-wise quadratic functions consistently give moderate accuracy, but accuracy improves more slowly as the indicator functions in Section 4.4 are added. We have begun developing a larger set of rational functions that shows some promise.
- The piece-wise quadratic functions do not scale well to larger networks because computing the regions S_k requires knowing the optimal policy for the fluid model. For larger networks, using the two-station fluid relaxation in [17] might be a tractable alternative, as discussed in Section 4.3.
- The constraint reduction method for piece-wise quadratic functions in Section 4.3 has not been tested. It would be interesting to test the speed and accuracy of ALP(N) compared to (ALPD).
- Performance of the policies recovered from the ALP remains a major open question. Numerical tests show that the policies can have poor performance or be unstable. However, using different approximation architectures or combining the ALP policies with heuristics such as safety stocks might lead to improved policies. It would be of interest to obtain bounds on their performance.

Appendix A Equivalence of LP form of the optimality equation

This appendix shows the equivalence of (1) to (LP) and (6). The argument is similar to, e.g., [20]. Using the constraints for the optimal policy and letting x_k denote the state after k transitions (including self-transitions

of the uniformized chain),

$$J \leq c'x_k + E_{u_*}[h(x_{k+1})|x_k] - h(x_k).$$

After summing and telescoping, taking expectations yields

$$J \leq \frac{1}{N} \sum_{k=0}^{N-1} E_{x, u_*} c'x_k + \frac{1}{N} E_{x, u_*} h(x_N) + \frac{1}{N} h(x_0).$$

We need to show that

$$\lim_{N \rightarrow \infty} \frac{1}{N} E_{x, u_*} h(x_N) = 0 \quad (36)$$

so that taking the limit as $N \rightarrow \infty$ leaves $J \leq J_*$ for any feasible h . Then, since (J_*, h_*) are feasible, they are optimal for (LP) and (6).

To show (36), use the fact that, for all policies u with finite $J(x, u)$,

$$\lim_{N \rightarrow \infty} \frac{1}{N} E_{x, u} |x_N|^2 = 0 \quad (37)$$

[12, Theorem 1]. Although they assume nonidling policies, (37) also holds for *weakly nonidling* policies where $u(t) \neq 0$ if $x(t) \neq 0$, which includes u_* . Their result also assumes x is in the recurrent class, but for the optimal policy this extends easily to all states. Combining (37) and (6) gives (36).

Appendix B ALP Constraints for the Series Queue

For the series queue of Section 4.2 and the differential cost approximation (19) this appendix gives the ALP constraints in terms of z , where $x = z + u$. The terms in (18) are

$$\begin{aligned} h(x + e_1) - h(x) &= q_{11}x_1 + q_{12}x_2 + \frac{1}{2}q_{11} + p_1 + r_2\beta^{x_2} \\ h(x - e_1 + e_2) - h(x) &= (-q_{11} + q_{12})x_1 + (q_{22} - q_{12})x_2 + \frac{1}{2}q_{11} + \frac{1}{2}q_{22} - q_{12} \\ &\quad - p_1 + p_2 - r_1(1 - \beta)\beta^{x_2} - r_2[(1 - \beta)x_1 + \beta]\beta^{x_2} - r_3[(1 - \beta)x_2 - \beta]\beta^{x_2} \\ h(x - e_2) - h(x) &= -q_{12}x_1 - q_{22}x_2 + \frac{1}{2}q_{22} - p_2 + r_1\left(\frac{\mu_1}{\mu_2} - 1\right)\beta^{x_2} \\ &\quad + r_2x_1\left(\frac{\mu_1}{\mu_2} - 1\right)\beta^{x_2} + r_3\left[\left(\frac{\mu_1}{\mu_2} - 1\right)x_2 - \frac{\mu_1}{\mu_2}\right]\beta^{x_2}. \end{aligned}$$

Using these, (18) can be written as (20), which we restate here

$$J \leq d^u + c^{uT}z + (\zeta^u + \xi^{uT}z)\beta^{z_2+u_2}.$$

For the control $u = (1, 1)$, $\xi^{(1,1)} = 0$ and

$$\begin{aligned} c_1^{(1,1)} &= c_1 - (\mu_1 - \alpha)q_{11} + (\mu_1 - \mu_2)q_{12} \\ c_2^{(1,1)} &= c_2 - (\mu_1 - \alpha)q_{12} + (\mu_1 - \mu_2)q_{22} \\ d^{(1,1)} &= c_1^{(1,1)} + c_2^{(1,1)} + \frac{1}{2}(\alpha + \mu_1)q_{11} - \mu_1 q_{12} + \frac{1}{2}(\mu_1 + \mu_2)q_{22} - (\mu_1 - \alpha)p_1 + (\mu_1 - \mu_2)p_2 \\ \zeta^{(1,1)} &= -r_2(\mu_2 - \alpha) - r_3(\mu_1 - \mu_2). \end{aligned}$$

For $u = (0, 1)$,

$$\begin{aligned} c_1^{(0,1)} &= c_1 + \alpha q_{11} - \mu_2 q_{12} \\ c_2^{(0,1)} &= c_2 + \alpha q_{12} - \mu_2 q_{22} \\ d^{(0,1)} &= c_2^{(0,1)} + \frac{1}{2}\alpha q_{11} + \frac{1}{2}\mu_2 q_{22} + \alpha p_1 - \mu_2 p_2 \\ \xi_1^{(0,1)} &= r_2(\mu_1 - \mu_2) \\ \xi_2^{(0,1)} &= r_3(\mu_1 - \mu_2) \\ \zeta^{(0,1)} &= \xi_2^{(0,1)} + r_1(\mu_1 - \mu_2) + r_2\alpha - r_3\mu_1. \end{aligned}$$

For $u = (1, 0)$, we must have $z_2 = 0$ and $\xi_1^{(1,0)} = 0$, $\zeta^{(1,0)} = 0$,

$$\begin{aligned} c_1^{(1,0)} &= c_1 - (\mu_1 - \alpha)q_{11} + \mu_1 q_{12} - r_2(\mu_1 - \mu_2) \\ d^{(1,0)} &= c_1^{(1,0)} + \frac{1}{2}(\alpha + \mu_1)q_{11} - \mu_1 q_{12} + \frac{1}{2}\mu_1 q_{22} - (\mu_1 - \alpha)p_1 + \mu_1 p_2 \\ &\quad - r_1(\mu_1 - \mu_2) - r_2(\mu_2 - \alpha) + r_3\mu_2. \end{aligned}$$

Finally, for $u = x = (0, 0)$, $\zeta^{(0,0)} = 0$ and

$$d^{(0,0)} = \frac{1}{2}\alpha q_{11} + \alpha p_1 + \alpha r_2.$$

Acknowledgements

Much of the numerical work in this paper was done by Jonathan Senning and by my students Lauren Berger, Taylor Carr, Lauren Carter, Jane Eisenhauer, Adam Elnagger, Jeff Fraser, Michael Frechette, Melissa LeClair, Josh Nasman, Chris Pfohl, and Nathan Walker; Anna Moore and Daniel Stahl also assisted. I would also like to thank Sean Meyn for his many suggestions. This work was supported in part by National Science Foundation grant 0620787.

References

- [1] D. Adelman. A price-directed approach to stochastic inventory/routing. *Operations Research*, 52(4):499–514, 2004.

- [2] F. Avram, D. Bertsimas, and M. Ricard. Fluid models of sequencing problems in open queueing networks: An optimal control approach. In F.P. Kelly and R.J. Williams, editors, *Stochastic Networks, Vol. 71 of the IMA Volumes in Mathematics and its Applications*, pages 199–234. Springer-Verlag, New York, 1995.
- [3] D. Bertsimas, D. Gamarnik, and J.N. Tsitsiklis. Performance of multiclass Markovian queueing networks via piecewise linear Lyapunov functions. *Ann. Appl. Probab.*, 11:1384–1428, 2001.
- [4] D. Bertsimas, I.Ch. Paschaladis, and J.N. Tsitsiklis. Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *Ann. Appl. Probab.*, 4:43–75, 1994.
- [5] R.-R. Chen and S.P. Meyn. Value iteration and optimization of multiclass queueing networks. *Queueing Systems Theory and Appl.*, 32(1-3):65–97, 1999.
- [6] D.P. de Farias and B. Van Roy. Approximate linear programming for average-cost dynamic programming. In *Advances in Neural Information Processing Systems 15*. MIT Press, 2003.
- [7] D.P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Oper. Res.*, 51(6):850–865, 2003.
- [8] D.P. de Farias and B. Van Roy. On constraint sampling for the linear programming approach to approximate dynamic programming. *Math. Oper. Res.*, 29(3):462–478, 2004.
- [9] D.P. de Farias and B. Van Roy. A linear program for Bellman error minimization with performance guarantees. In *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.
- [10] B. Hajek. Optimal control of two interacting service stations. *IEEE Trans. Automat. Control*, AC-29:491–499, 1984.
- [11] J.M Harrison. Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete-review policies. *Ann. Appl. Probab.*, 8(3):822–848, 1998.
- [12] P.R. Kumar and S.P. Meyn. Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies. *IEEE Trans. Automat. Control*, 41(1):4–17, 1996.
- [13] S. Kumar and P. R. Kumar. Performance bounds for queueing networks and scheduling policies. *IEEE Trans. Automat. Control*, 39:1600–1611, 1994.
- [14] S.P. Meyn. The policy iteration algorithm for Markov decision processes with general state space. *IEEE Trans. Automat. Control*, AC-42:191–197, 1997.
- [15] S.P. Meyn. Sequencing and routing in multiclass queueing networks. Part I: Feedback regulation. *SIAM J. Control Optim.*, 40:741–776, 2001.
- [16] S.P. Meyn. Stability, performance evaluation and optimization. In E. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer, 2001.

- [17] S.P. Meyn. Sequencing and routing in multiclass queueing networks. Part II: Workload relaxations. *SIAM J. Control Optim.*, 42(1):178–217, 2003.
- [18] S.P. Meyn. Dynamic safety-stocks for asymptotic optimality in stochastic networks. Dept. of Electrical and Computer Eng., University of Illinois at Urbana-Champaign, 2004.
- [19] C.C. Moallemi, S. Kumar, and B. Van Roy. Approximate and data-driven dynamic programming for queueing networks. Working paper, Graduate School of Business, Columbia University, 2006.
- [20] J.R. Morrison and P.R. Kumar. New linear program performance bounds for queueing networks. *J. Optim. Theory Appl.*, 100(3):575–597, 1999.
- [21] J.R. Morrison and P.R. Kumar. Computational performance bounds for Markov chains with applications. *IEEE Trans. Autom. Control*, 53:1306–1311, 2008. Full-length version available at <http://black.csl.uiuc.edu/~prkumar>.
- [22] C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of optimal queueing network control. *Math. Oper. Res.*, 24:293–305, 1999.
- [23] I.C. Paschaladis, C. Su, and M.C. Caramanis. Target-pursuing scheduling and routing policies for multiclass queueing networks. *IEEE Trans. Automat. Control*, 49(10):1709–1722, July 2004.
- [24] M. C. Russell, J. Fraser, S. Rizzo, and M. H. Veatch. Comparing LP bounds for queueing networks. To appear, *IEEE Trans. Autom. Control*, 2009.
- [25] D. Schuurmans and R. Patrascu. Direct value-approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1579–1586. MIT Press, 2001.
- [26] P. Schweitzer and A. Seidmann. Generalized polynomial approximations in Markovian decision processes. *J. of Mathematical Analysis and Applications*, 110:568–582, 1985.
- [27] L.I. Sennott. *Stochastic Dynamic Programming and the Control of Queueing Systems*. Wiley, New York, 1999.
- [28] H. Topaloglu and S. Kunnunkal. Approximate dynamic programming methods for an inventory allocation problem under uncertainty. *Naval Research Logistics*, 53:822–841, 2006.
- [29] M. H. Veatch and N. Walker. Approximate linear programming for network control: Column generation and subproblems. Working paper, Gordon College, Dept. of Math. Available at <http://faculty.gordon.edu/ns/mc/mike-veatch>, 2008.
- [30] M.H. Veatch. Fluid analysis of arrival routing. *IEEE Trans. Automat. Control*, 46:1254–1257, 2001.
- [31] M.H. Veatch. Using fluid solutions in dynamic scheduling. In S. B. Gershwin, Y. Dallery, C. T. Papadopoulos, and J. M. Smith, editors, *Analysis and Modeling of Manufacturing Systems*, pages 399–426, New York, 2002. Kluwer.
- [32] G. Weiss. On optimal draining of fluid reentrant lines. In F.P. Kelly and R.J. Williams, editors, *Stochastic Networks, Vol. 71 of the IMA Volumes in Mathematics and its Applications*, pages 93–105, New York, 1995. Springer-Verlag.