Computers & Operations Research 63 (2015) 32–45

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Approximate linear programming for networks: Average cost bounds

Michael H. Veatch *

Department of Mathematics, Gordon College, Wenham, MA 01984, USA

ARTICLE INFO

Available online 1 May 2015

Keywords:
Queueing network
Approximate dynamic programming
Linear programming

ABSTRACT

This paper uses approximate linear programming (ALP) to compute average cost bounds for queueing network control problems. Like most approximate dynamic programming (ADP) methods, ALP approximates the differential cost by a linear form. New types of approximating functions are identified that offer more accuracy than previous ALP studies or other performance bound methods. The structure of the infinite constraint set is exploited to reduce it to a more manageable set. When needed, constraint sampling and truncation methods are also developed. Numerical experiments show that the LPs using quadratic approximating functions can be easily solved on examples with up to 17 buffers. Using additional functions reduced the error to 1–5% at the cost of larger LPs. These ALPs were solved for systems with up to 6–11 buffers, depending on the functions used. The method computes bounds much faster than value iteration. It also gives some insights into policies. The ALPs do not scale to very large problems, but they offer more accurate bounds than other methods and the simplicity of just solving an LP.

1. Introduction

Queueing networks are a common modeling framework for manufacturing, computer, and communication systems. Even under the simple assumptions of exponentially distributed service and interarrival times, a multiclass network (MQNET) structure, linear holding costs, and sequencing and routing control, the optimal control problem is NP-hard so that we cannot hope to solve large instances exactly [1]. In fact, only examples with a handful of buffers have been solved, particularly in heavy traffic. This paper develops approximate linear programming (ALP) algorithms for these problems. A sequence of lower bounds on average cost are computed by solving LPs with an increasing number of variables. Policies obtained from the ALP are also considered.

A practical use of the lower bounds is benchmarking the many heuristic policies that have been proposed. To be useful for benchmarking, the bounds must be within a few percent of optimal. The candidate policy can then be simulated to estimate its average cost. If the difference between the lower bound and candidate policy is small, we may conclude that the policy is close to optimal. Available bounds [2,3] are generally not that accurate. Other justifications of heuristic policies such as stability or showing asymptotic optimality under heavy traffic or fluid scaling do not measure suboptimality. Motivated by the need for benchmarking, we develop ALP bounds that are more accurate than prior bounds, at the expense of more computation, and are computable for moderate-sized networks.

The first contribution of this paper is identifying good approximating functions for the differential cost. All approximate dynamic programming (ADP) methods must choose a compact approximation architecture, which is crucial to the accuracy of the method. For queueing networks, Refs. [4,5] use quadratic and cubic approximations, Ref. [6] uses functions of one variable, and Ref. [7] uses linear functions. In our tests on average cost problems, these architectures often gave inaccurate lower bounds. Various analyses and numerical testing led to the following types of functions: (i) quadratics, (ii) exponential decay, (iii) piece-wise quadratics, with the regions taken from the associated fluid model, (iv) rationals, (v) piece-wise quadratics on a variable grid, and (vi) functions of one or two variables. Compared to earlier studies and other functions we tested, these functions offer a better trade-off between accuracy and the number of functions. By judiciously adding some of them to the ALP, errors of 1–5% were achieved on all test problems where optimality could be computed. The number of functions needed to achieve these small errors appears to be exponential in the number of buffers, which is to be expected given the difficulty of the problem. In comparison, the traditional method of using dynamic programming on a truncated state space requires orders of magnitude more states than the ALP method requires functions for the same accuracy. Thus, the approximation architectures are more compact than the original problem, but not scalable to large systems.

Several factors influenced the choice of approximating functions. Queueing network problems suffer from two of the three "curses of

* Tel.: +1 978 867 4375.
E-mail address: mike.veatch@gordon.edu

http://dx.doi.org/10.1016/j.cor.2015.04.014
0305-0548/

dimensionality" described in [8]: large state spaces and a large number of actions. The number of transitions, however, is small, making the resulting LP sparse for certain approximating functions. The functions (i), (ii), (iii), and (vi) are also appealing because they allow some degree of constraint reduction, described below. Quadratic functions, and particularly the piece-wise quadratics (iii), are motivated by considering the associated fluid model.

The second contribution of this paper addresses solving the ALPs, which contain one variable for each approximating function and one constraint for each state-action pair. To create an LP with a manageable number of constraints, several authors use constraint sampling. Although this method has theoretical support [9] and is used on the game of tetris in [7], it has serious limitations in our numerical tests. As an alternative, we provide new methods of reducing the number of constraints that exploit their structure. The number of reduced constraints varies, but is always proportional to the number of actions, which for many networks is exponential in the number of buffers. To accommodate general functions and larger problems, we also use constraint sampling and hybrid approaches.

We also briefly address the *policies* associated with the differential cost approximations. We report some cases where the ALP gives a useful policy and relate the approximation architecture to the form of policy obtained. A final contribution is that we relate error in the differential cost approximation, as measured by expected Bellman error, to the accuracy of the ALP average cost.

The method applies to a broad class of queueing control problems, modeled as Markov decision processes (MDPs). Tests included problems with reentrant flow, arrival routing, probabilistic routing, and cross-trained servers. Accuracy was tested on networks with up to six buffers, where the optimal solution could be computed. ALPs were also solved for an 11-buffer manufacturing network and series lines with up to 17 buffers. These tests used software that does not include all of the constraint reduction methods. Significant speed-ups could be achieved by implementing more constraint reduction or using customized LP algorithms as in [7]. However, our software has the advantage of being very general and using commercial LP solvers. The general approach should be useful for other MDPs on high-dimensional state spaces.

An average cost objective is used for several reasons. In many applications, a long time horizon is more realistic and avoids having to choose a discount rate. Furthermore, the optimality equations can be written using difference operators, facilitating constraint reduction. The average cost problem can also be related to the fluid model, giving some guidance in the choice of approximating functions.

The ALP approach was originally proposed in [10]. For discounted MDPs on a finite state space, Refs. [11,7] 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 the space of approximating functions, 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. Constraint sampling is shown to be probabilistically accurate in [9]. Bounds for average cost problems are given in [4,12,13]. In [14], column generation methods are used to solve average cost ALPs more efficiently. Constraint reduction for quadratic and piece-wise quadratic functions is used in [15,16]. They consider a different quadratic on each set of states defined by which buffers are empty. We extend this method to consider the piece-wise quadratic functions (iii), which are defined on affine sets of states. We also develop a new method of reducing constraints for certain exponential functions. Our constraint reduction for (vi) is based on the notion of factored value functions and MDPs, introduced in [17]. Constraint reduction for factored problems is used in [18–20].

Given a set of approximating functions, many ADP methods have been proposed for approximating the value function as a linear combination of these functions. However, simulation-based methods, such as least squares temporal difference, tend to require customization to the specific problem. See [21, Chapter 6] for a survey and [8] for detailed coverage. We focus on the ALP approach because of the approximation guarantees described above and the simplicity of just solving an LP. The smoothed ALP method of [7] seeks to improve the accuracy of the ALP with discounted cost through a Lagrangian relaxation. Promising results are given for a small queueing network, but only linear approximating functions are tested. The information relaxation, or martingale duality, approach [22,23] has been applied to inventory control and option pricing. It solves a deterministic version of the problem, where future random events are known, for repeated simulation paths. The method requires an estimate of the value function. Optimizing over all value functions in an approximation architecture leads to a convex "outer" optimization problem [24,25]. As noted in the last reference, this dual method dominates the ALP bound (for discounted cost problems), however, it is only tractable when the deterministic inner problem has a simple structure, which is not true of queueing networks.

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 the various approximating functions are introduced. Constraint reduction for some of these functions is presented in Section 5. Numerical results are presented in Section 6, and Section 7 concludes.

2. Open MQNET sequencing

In this section we describe the standard MQNET model and the fluid model associated with it. The results in Sections 3 and 5 can be extended to more general stochastic processing networks; the examples in Section 6 include the additional features of arrival routing and servers with overlapping job classes. 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. If, in addition, routes do not merge then either $p_{p(i),i}=1$, where $p(i)$ is the unique predecessor of class i, or i has no predecessor.

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., $I+P+P^2+\cdots$ 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

$$\rho=C\ \mathrm{diag}(m_1,\ldots,m_n)\lambda$$

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

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

$$Cu(t) \le \mathbf{1}$$
$$u_i(t) \le x_i(t).$$

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

The objective is to minimize long-run average cost

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

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

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

Let $\mathcal{A}(x)$ be the set of feasible 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
$$J+h(x) = c'x + \min_{u\in\mathcal{A}(x)} (P_u h)(x). \tag{1}$$
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
$$h_*(x) = \limsup_{T\to\infty} \left(E_{x,u_*} \int_0^T c'x(t)\,dt - E_{0,u_*} \int_0^T c'x(t)\,dt \right). \tag{2}$$

Properties (1) and (2) can be established using general results for MDPs as in [26, Theorems 7.2.3 and 7.5.6]. For networks, properties (1) and (2) are shown in [27, Theorem 7]; (3) is obtained by applying standard verification theorems to networks; see, e.g., [28, Theorem 2.1 and Section 7] and [29, 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. The fluid control problem corresponding to (1) is

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

where $\alpha=(\alpha_1,\ldots,\alpha_n)'$ and $B=(P'-I)\,\mathrm{diag}(\mu_1,\ldots\cdot\mu_n)$. For each initial state the time horizon T is chosen such that $q(t)=0$ for all $t\ge T$. The fluid cost $V(x)$ guides some of our approximations of $h(x)$. The motivation for this approximation is [27, Theorem 7 (iv)], based on [28, Theorem 5.2]. It establishes the following connection between the stochastic and fluid cost functions:

$$\limsup_{\|x\|\to\infty} \frac{|h_*(x)-V(x)|}{\|x\|^2} = 0. \tag{3}$$

See [30] for discussion of the policy implications. A scaling argument shows that V grows quadratically in any direction: $V(\theta x)=\theta^2 V(x)$. Motivated by examples, we assume that this quadratic structure consists of finitely many "pieces," i.e.,

$$V(x) = \tfrac{1}{2}x'Q_k x, \quad x\in S_k \tag{4}$$

for some partition $S_1,\ldots,S_K$ of $\mathbf{R}_+^n$. Combining (3) and (4)

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

The scaling property of V implies that each S_k can be chosen to be a cone, i.e., if S_k contains x, it also contains θx, $\theta>0$. Because V is continuous, we can assume that each S_k has full dimension. The optimal fluid policy also inherits this property if ties are broken appropriately. Hence, the *control switching sets* where the control is constant are also cones. We further assume that each S_k is convex and polyhedral (but are not aware of a proof).

3. Approximate LP: average cost bounds

This section describes the approximate linear program which approximates the differential cost and places a lower bound on average cost. We also show that the error in this bound is related to Bellman error. For problems with finite state spaces and J_* independent of the initial state, an inequality relaxation of Bellman's equation gives an equivalent LP in the same variables (see, e.g., [21])

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

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

$$-L_1 \le h(x) \le L_2(1+|x|^2). \tag{6}$$

The equivalence is shown in [15]; see [31, Prop. 9.2.11] for a general MDP setting.

This *exact LP* has one variable for every state. To create a tractable LP, the differential cost can be approximated by a linear form

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

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

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

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 formed by adding the constraints $h=\Phi r$ to the exact LP, the exact LP is a relaxation. Hence, (ALP) gives a lower bound, $J_*\ge\underline{J}$. (ALP) is feasible, so it has an optimal solution, say r_*. By reversing the first inequality in (ALP) and restricting the class of policies, one can compute an upper bound on the average cost of these policies. This upper bound is used to check stability of, say, all nonidling policies.

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.

It is not obvious that (ALP) will provide a good approximation Φr_* to the differential cost h_*. We use (ALP) primarily to estimate J_*. However, the proposition below provides some connection between approximating h_* and approximating J_*. Given any J and h, Bellman error is defined as

$$B(x) = \min_{u \in \mathcal{A}(x)} (P_u h)(x) - h(x) + c'x - J. \tag{8}$$

If J, h satisfy the constraints (ALP) then $B(x) \geq 0$.

Proposition 1. *For any J, r, let $h = \Phi r$ and let $\tilde{u}$ be an h-greedy policy. In addition to (6), 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

$$J = J_{\tilde{u}} - E_{\tilde{u}}(B). \tag{9}$$

Proof. Since $\tilde{u}$ achieves the minimum in (8)

$$B(x) = (P_{\tilde{u}} h)(x) - h(x) + c'x - J. \tag{10}$$

Taking expectations

$$\begin{aligned} E_{\tilde{u}}(B) &= E_{\tilde{u}}[(P_{\tilde{u}} h)(x) - h(x) + c'x - J] \\ &= E_{\tilde{u}}[c'x - J] \\ &= J_{\tilde{u}} - J. \end{aligned}$$

The crucial second equality above holds by Proposition 8.2.5 of [31], which requires A2 and the growth condition (6).□

The significance of (9) is that

$$J_* - J = E_{\tilde{u}}(B) - (J_{\tilde{u}} - J_*). \tag{11}$$

In words, the average cost error decomposes into the expected Bellman error under this policy minus its suboptimality. Given an approximation architecture for which the suboptimality is relatively small, this suggests a criteria for selecting functions to add to the approximation: They should address Bellman error in states with a large contribution to its expectation.

4. Differential cost approximation

This section defines and motivates the functions ϕ_k used to approximate the differential cost.

(i) *Quadratic functions*

A quadratic approximation will be written

$$h(x) = \tfrac{1}{2} x' Q x + px \tag{12}$$

where $Q = [q_{ij}]$ is symmetric. For convenience, the variable names q_{ij} and p_i will be kept rather than mapping them into r_k.

(ii) *Exponential functions*

Graphs of h_* and the importance of states with small x_i led us to use the functions

$$\phi_i(x) = \beta_i^{x_i} \tag{13}$$

$$\phi_{ij}(x) = x_j \beta^{x_i} \tag{14}$$

where $\beta_i < 1$. Greedy policies for the approximation (13) include more realistic policies than the linear switching curve policies that result when h is quadratic. For example, consider adding just $r\phi_2$ to the quadratic (12) for the series queue. Server 1 is busy in the greedy policy when

$$\begin{aligned} h(x - e_1 + e_2) - h(x) &= (q_{12} - q_{11})x_1 + (q_{22} - q_{12})x_2 + \tfrac{1}{2}(q_{11} + q_{22}) \\ &\quad - q_{12} - p_1 + p_2 - r(1-\beta)\beta^{x_2} < 0. \end{aligned} \tag{15}$$

Solving this ALP numerically often leads to $q_{22} = q_{12}$ and $r_1 < 0$; then (15) reduces to

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

for some A and B. Numerical experience and [32] suggest a logarithmic form to the optimal switching curve. Thus, the approximation architecture has the potential to produce realistic switching curves; see Section 6.4.

(iii) *Piece-wise quadratic on cones*

A piece-wise quadratic function on the same regions as the fluid cost (4) is

$$h(x) = \tfrac{1}{2} x' Q^k x + (p^k)' x + f^k, \quad x \in S_k. \tag{16}$$

The regions S_k are found in Section 5.3 for an example with three classes. For large problems, finding the fluid policy or $\{S_k\}$ becomes intractable but heuristics could be used to partition the state space.

(iv) *Rational functions*

Typically, higher-order terms would be added to a quadratic approximation. However, the functions need to be bounded by a quadratic, so we divide by a suitable polynomial of degree two less than the numerator. Also, instead of using different powers, i.e., $1, x_i, x_i^2$, we use $f_{i,1}(x_i) = x_i^2$, $f_{i,2}(x_i) = x_i(N_i - x_i)^+$, $f_{i,3}(x_i) = [(N_i - x_i)^+]^2$, where $x^+ = \max\{x, 0\}$. Note that $f_{i,2}$ and $f_{i,3}$ are zero beyond N_i, emphasizing the interval $0 \leq x_i \leq N_i$ and increasing the sparsity of the constraint matrix. The numerical tests in Section 6 use $N_i = 6/(1 - \rho_{\sigma(i)})$. The three choices for j lead to 3^n rational functions

$$\frac{\prod_{i=1}^n f_{i,j(i)}(x_i)}{\left(1 + \sum_{i=1}^n x_i/N_i\right)^{2(n-1)}}, \quad j(i) = 1, 2, 3. \tag{17}$$

(v) *Piece-wise quadratic on rectangular regions*

A common function approximation is to use a separate polynomial on different regions, as done in spline functions or local polynomials. Instead of a uniform grid, we define rectangular regions that grow exponentially as queue length grows. Let

$$S(y) = \left\{ x : 2^{y_i - 1} - \tfrac{1}{2} \leq x_i < 2^{y_i}, i = 1, \ldots, n \right\} \tag{18}$$

for all $y \in Z_+^n$ such that $y \neq 0$ and $0 \leq y_i < M$ for some M. These $n^M - 1$ sets cover the states (excluding the origin) with $x_i < 2^{M-1}$, dividing each x_i into the sets $\{0\}, \{1\}, \{2, 3\}, \{4, 5, 6, 7\}$, etc. The piece-wise quadratic approximation on these sets is

$$h(x) = \tfrac{1}{2} x' Q^y x + (p^y)' x + f^y, \quad x \in S(y). \tag{19}$$

The number of basis functions, or variables, in (19) is large, roughly $n^{M-2} n^2/2$, after eliminating unneeded variables. For example, if $y_i = 0$ or 1, then we can set $q_{ij}^y = p_i^y = 0$ because there is only one value of x_i in $S(y)$. The numerical results in Section 6 also modify (18) by removing the upper bound when $y_i = M - 1$, so that the sets cover all of Z_+^n. With this modification, $M = 2$ corresponds to the functions in [15]. We also consider the piece-wise linear approximation on these regions.

(vi) *Functions of one or two variables*

Consider a differential cost approximation of the form

$$h(x) = \sum_{i=1}^n h_i(x_i) \tag{20}$$

using functions of one variable. To parameterize h_i, use the indicator functions $\phi_j(x_i) = 1_{\{x_i = j\}}$ and let

$$h_i(x_i) = \sum_{j=0}^{M-1} r_{ij}\phi_j(x_i) + r_{i0}\phi_0(x_i). \tag{21}$$

Note that $h_i(x)$ is arbitrary for $x < M$ and has the form ϕ_0 for $x \geq M$. This parameterization is used in [6] with $\phi_0(x_i) = x_i \ln x_i$, which we include in our tests. The number of functions in (20) is linear in the number of classes, so that it is suitable for large problems. We use an ad hoc modification of (20) to improve accuracy: replace the indicator functions ϕ_j with

$$\phi_j(x) = 1_{\{x_i = j\}}\left(1 + \gamma|x_{(i)}|^2\right)B^{|x_{(i)}|} \tag{22}$$

where $|x_{(i)}| = \sum_{k \neq i} x_k$ and $\gamma \geq 0$, $B \leq 1$ are parameters to be chosen. Setting $\gamma = 0$ and $B = 1$ recovers the indicator functions.

Our numerical tests also use functions of a single workload, proposed in [6]. Define the workload $w_j(t)$ of station j as the expected remaining time that it will spend serving jobs currently in the system. The mapping from queue length to workload is $w(t) = CM^{-1}(I - P')^{-1}x(t)$, where $M = \text{diag}(\mu_i)$ and C is the constituency matrix from Section 2. Because w is not integer, a different parameterization is needed. We use a piece-wise constant $h_j(w_j)$, with M intervals of width δ plus $\phi_0(w_j) = w_j \ln w_j$; δ is set experimentally.

Finally, we consider functions of two variables. For a series or reentrant line, we use the pair of variables x_i and $\sum_{j=i+1}^{n} x_j$, for $i = 1, \ldots, n-1$, known as the echelon inventory. These two variables are commonly used in control policies in manufacturing; see [33].

5. Constraint reduction

For certain approximating functions, the constraints of the resulting ALP can be algebraically reduced to a smaller, or at least more easily approximated, set. The reductions achieved are summarized in Table 1, which references the equation specifying the functions and the section where the reduction is presented. Without truncation, the number of constraints is infinite. The standard DP approach is to truncate the state space to $x_i < N$. The truncated problem contains roughly $N^n|\mathcal{A}|$ constraints, where $|\mathcal{A}|$ is the number of actions. Here $\mathcal{C}$ is a reduction in dimension that depends on the problem structure, κ is the number of cones, and $|\Psi|$ is the number of regions defined in Section 5.3; there are no more than $\kappa|\mathcal{A}|$ regions of full dimension and this can be used as a rough guide for $|\Psi|$. As explained in this section, the N used for exponential and piecewise quadratic functions can be set much smaller than in DP.

The reduced constraints achieve better dependence on the truncation – sometimes much better. However, the number of constraints is still proportional to the number of actions, which is generally exponential in n. If nonidling is assumed, the number of constraints can be estimated by counting only the actions where all servers are busy. Finally, constraint reduction is not always available when multiple types of functions are used; exponential and functions of one variable can be combined, as can quadratic, exponential, and functions of two variables.

Table 1
Approximate number of reduced constraints.

Type of function	Approximate number of	
	Functions	Constraints
Quadratic (12) 5.1	$n^2/2$	$(n+1)\|\mathcal{A}\|$
Exponential (13) 5.2	$n^2/2$	$N^{n-\mathcal{C}}$
Two-class series queue App. A	8	$2N+8$ vs. $3N^2$
Three-class reentrant line 5.2	14	$O(N)$ or $O(N^2)$ vs. $6N^3$
Piecewise quad. on cones ALP(N) (16) 5.3	$\kappa n^2/2$	$N^n\|\mathcal{A}\| + n\|\Psi\|$
Functions of one variable (20) 5.4 (only eliminating x; 2nd term is for series line)	nN	$\|\mathcal{A}\| + (4n-4)N$
Functions of two variables 5.4 (overlapping pairs)	nN^2	$\|\mathcal{A}\| + 8nN^2$

5.1. Quadratic functions

The constraints (ALP) can be reduced to a finite set for quadratic h [15, Appendix A]. To simplify notation, consider only deterministic routing. First, we write the constraints as

$$J \leq \sum_{i=1}^{n}(c_i x_i + \alpha_i[h(x+e_i) - h(x)] + u_i\mu_i[h(x - e_i + e_{s(i)}) - h(x)]) \quad \text{for all } x \in \mathbf{Z}_+^n,\ u \in \mathcal{A}(x). \tag{23}$$

Unlike a discounted model, only differences in h appear in these constraints, simplifying the analysis. It is convenient to use the substitution $x = z + u$, so that a control u is feasible for all $z \in Z_+^n$. Substituting (12) into (23) yields

$$J \leq d^u + (c^u)'z \quad \text{for all } z \in \mathbf{Z}_+^n,\ u \in \mathcal{A} \tag{24}$$

where

$$c_i^u = c_i + \sum_{j=1}^{n}[\alpha_j q_{ij} + u_j\mu_j(q_{i,s(j)} - q_{ij})]$$

$$d^u = \sum_{i=1}^{n}\left[u_i\left(c_i^u + \mu_i\left(\frac{1}{2}q_{ii} + \frac{1}{2}q_{s(i),s(i)} - q_{i,s(i)} + p_{s(i)} - p_i\right)\right) + \alpha_i\left(\frac{1}{2}q_{ii} + p_i\right)\right]$$

and $c^u = [c_i^u]$. For (24) to hold for all z, the right hand side must be nondecreasing in z_i. Hence, (24) is equivalent to

$$J \leq d^u \quad \text{for all } u \in \mathcal{A} \tag{25}$$

$$c_i^u \geq 0 \quad \text{for } i = 1, \ldots, n,\ u \in \mathcal{A}. \tag{26}$$

If the optimal policy is nonidling, then for a given control u, (26) 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 $n+1$.

5.2. Exponential functions

Constraint reduction for quadratic functions relied on the fact that the constraints are linear in x. Depending on which exponential functions are used, constraints will be linear in some x_i and uni-min in others, allowing some constraint reduction and efficient truncation. Assume that each class has at most one predecessor, i.e., routing is deterministic and routes do not merge, and that there are no arrivals at classes that have a predecessor class. If class i has no predecessor, set $p(i) = 0$. Consider the approximation

$$h(x) = \frac{1}{2}x'Qx + px + \sum_{i:p(i)\neq 0} r_i\beta_i^{x_i}$$

and assume $\beta_i = \mu_i/\mu_{p(i)} < 1$.

The constraints (23) are

$$J \leq d^u + (c^u)'z + \sum_{i:p(i)\neq 0} r_i(u_{p(i)} - u_i)(\mu_i - \mu_{p(i)})\beta_i^{z_i + u_i} \quad \text{for all } z \in \mathbf{Z}_+^n,\ u \in \mathcal{A} \tag{27}$$

where d^u, c^u, and $z=x-u$ are from Section 5.1. Consider (27) for a given action u and let $\mathcal{C}(u)$ be the set of classes i for which there is no $\beta_i^{z_i}$ term. For (27) to hold for all z, we must have $c_i^u \geq 0$ for $i \in \mathcal{C}(u)$. Given these constraints, (27) is only needed at $z_i=0$, $i \in \mathcal{C}(u)$, where it is tightest. Hence, the constraints are equivalent to $c_i^u \geq 0$ for $i \in \mathcal{C}(u)$ and (27) for all $z \in \mathbf{Z}_+^n$ such that $z_i=0$, $i \in \mathcal{C}(u)$.

This set of constraints is still infinite, but in fewer dimensions. The reduction in dimension, $\mathcal{C}=\min_{u\in\mathcal{A}}|\mathcal{C}(u)|$, is at least the number of classes with no predecessor and usually is larger if nonidling is assumed. We will approximate (ALP) using the constraints $c_i^u \geq 0$ for $i \in \mathcal{C}(u)$ and (27) at $z_i=0$, $i \in \mathcal{C}(u)$ and $z_i=0,\ldots,N-1$ for $i \notin \mathcal{C}(u)$ for some N. Call this relaxation ALP(N). The total number of constraints is $\sum_{u\in\mathcal{A}}(|\mathcal{C}(u)|+N^{n-|\mathcal{C}(u)|})$. In our experiments a very small truncation often suffices – the solution to ALP(N) is the same for all $N \geq M$ for some small M. This behavior appears to be due to the fact that if the sign of r_i is such that the coefficient of the $\beta_i^{z_i}$ term is positive, then (27) is uni-min in z_i and the minimum typically occurs at small z_i.

Constraint reduction is also possible when some of the functions (14) are used. Appendix A presents a reduction to a one-dimensional set for the two-class series queue. For the three-class reentrant line in Fig. 1, the analogous h approximation includes (13) and (14) for $i=2, 3$. The constraint set reduces to a two-dimensional set (one-dimensional if only the functions (13) are used).

5.3. Piece-wise quadratic on cones

This section presents two approximate constraint reductions for the piece-wise quadratic functions (16), one of which extends the dual method in [16, Section 34], where they consider the rectangular regions (18) with $M=2$. The underlying fluid analysis is illustrated on a three-class reentrant line. First, we construct sets of states in which the constraints (23) are quadratic for a given u. We must account for transitions between the quadratic regions $\{S_k\}$. Let $\{\delta_l\}$, $l=1,\ldots,q$ be the change in the state for each possible transition, e.g., $\delta_1=-e_1+e_2$ for a class 1 service completion that moves to class 2. Introduce an indexing vector $\psi=(u,k_0,k_1,\ldots,k_q)$ and define

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

Let $\Psi=\{\psi : X^\psi\neq\emptyset\}$. There is one index ψ for each combination of action 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 $k_l=k_0$. Note that $\{X^\psi\}$ contains one partition of the state space for each action and that these partitions are finer than $\{S_k\}$.

Let $x=z+u$ and $Z^\psi=\{z : z+u\in X^\psi\}$. For a given u and region X^ψ, the constraints are quadratic:

$$J\leq d^\psi+(c^\psi)'z+\tfrac{1}{2}z'M^\psi z \quad \text{for } z\in Z^\psi \text{ for all } \psi \tag{28}$$

where d^ψ, c^ψ, and M^ψ are linear functions of $\{Q^k, p^k, f^k, k=1,\ldots,\kappa\}$. The quadratic term M^ψ is symmetric. It appears because of transitions between regions S_k. First, we remove the integer restriction by allowing $z\in\overline{Z}^\psi$, where $\overline{Z}^\psi$ is a polyhedron, say $\{z\in R^n : A^\psi z\geq b^\psi, z\geq 0\}$, whose lattice points are Z^ψ. One could also approximate $\overline{Z}^\psi$ with a larger polyhedra using fewer constraints.

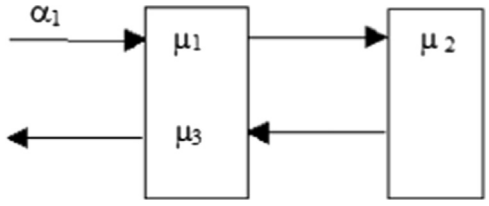

Fig. 1. Three-class, two-station reentrant line.

The dual reduction method replaces (28) with the stronger, simpler conditions

$$M^\psi\geq 0 \tag{29}$$

and

$$J\leq d^\psi+(c^\psi)'z \quad \text{for all } z \text{ such that } A^\psi z\geq b^\psi \text{ and } z\geq 0. \tag{30}$$

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 (30) 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

$$\begin{aligned}\max\quad & (b^\psi)'y^\psi\\ \text{s.t.}\quad & (A^\psi)'y^\psi\leq c^\psi\\ & y^\psi\geq 0.\end{aligned} \tag{31}$$

Thus, (31) and $w^\psi\geq J-d^\psi$ for all ψ are equivalent to (30). Reintroducing J, Q^k, p^k, and f^k as variables, the dual form of (ALP) is

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

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 more than $n^2/2$ constraints for each ψ. It contains the roughly $\kappa n^2/2$ variables in (16) (recall that κ is the number of quadratic regions) plus one dual variable for each constraint used to define the polyhedra $\overline{Z}^\psi$. 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. We propose a second linear approximation that avoids the dual variables.

Recall that $\{S_k\}$ are convex polyhedral cones from the origin, so $\overline{Z}^\psi$ is unbounded. Call its extreme directions $\{\beta^{\psi,l}\}$. A better approximation than (29) is $(\beta^{\psi,l})'M^\psi\beta^{\psi,l}\geq 0$ for all ψ and directions $\beta^{\psi,l}$. This constraint, combined with $(c^\psi)'\beta^{\psi,l}\geq 0$, guarantees that (28) holds in all extreme directions. If we also required (28) at the extreme points, we should have a good approximation of (28) on $\overline{Z}^\psi$ (approximate because it is not linear for some ψ). Finding the extreme directions is made easier by the fact that the extreme directions of $\overline{Z}^\psi$ are a subset of the extreme directions of S_{k_0}. In particular, $\overline{Z}^\psi$ 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 $\overline{Z}^\psi$). Rather than computing all the extreme points, we will include (28) at points near the origin, say, $z_i<N$, in Z^ψ. This approximation should be accurate for small N because the hyperplanes bounding each S_k pass through the origin, and those bounding $\overline{Z}^\psi$ are close to them in a certain sense: if any $k_l\neq k_0$, then all points in Z^ψ are within one transition of a hyperplane separating S_{k_l} and S_{k_0}, so $\overline{Z}^\psi$ lies in a thickened boundary. From the geometry of $\{S_k\}$ it can be argued that hyperplanes of $\overline{Z}^\psi$ lie close to the origin. Conversely, if all transitions from Z^ψ are to points in S_{k_0}, then $\overline{Z}^\psi$ is essentially S_{k_0} with thickened boundaries removed, which also has extreme points near the origin.

Combining the points near the origin and the extreme directions, the approximation ALP(N) contains the constraints

(28) for $z\in Z^\psi$ and $z_i\leq N-1$

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

$$(\beta^{\psi,l})'M^{\psi}\beta^{\psi,l} \geq 0 \tag{33}$$

for all ψ and all directions $\beta^{\psi,l}$. ALP(N) has two limiting constraints for every extreme direction $\beta^{\psi,l}$, which is generally exponential in n, plus N^n constraints (28).

To illustrate these definitions, consider the three-class, two-station reentrant line in Fig. 1. 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 [34] 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 has $\kappa = 3$ quadratic regions, depending on which of three actions will be used next on a trajectory starting from x. The correspondence between quadratic regions and control switching sets is shown in Table 2. One can verify that these are the three quadratic regions by following the trajectories. Trajectories in S_1 have $\dot{q} = (\alpha, 0, \mu_2 - \mu_3)$ and enter S_2 next; S_2 and S_3 feed into the switching set $x_2 = 0$ and $x_3 = 0$, which leads to $x = 0$. Note that the optimal policy in S_3 is not unique; a nonidling optimal policy also exists. The extreme directions of the quadratic regions are listed in Table 3.

To illustrate the definition of X^{ψ}, 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., $x \in S_3$. 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. Although there are $\kappa^{q+1}|\mathcal{A}|$ indices ψ, the number $|\Psi|$ of nonempty regions is much smaller. In this example, $|\mathcal{A}| = 6$ and $q = 4$, giving 1458 values of ψ. If nonidling is assumed, there are at most two actions in each state and there are only six regions of full dimension (two actions in each S_k) and a modest number of smaller regions.

Solving the fluid model to find $\{S_k\}$ and constructing $\{\overline{Z}^{\psi}\}$ become difficult for larger problems with more regions. One alternative is to analyze the two-station fluid workload relaxation in [35], for which an optimal policy can easily be found and translated back to the original problem. The quadratic regions for this policy could be determined systematically by working backward from the origin and determining all sequences of control regions that can be visited.

Table 2
Quadratic regions of the fluid cost in the reentrant line example.

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

Table 3
Edges of the quadratic regions in the reentrant line example.

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	$(1, 0, 0)$, $(0, 1, 0)$, $(0, \mu_2, \alpha\gamma + \mu_3 - \mu_2)$

5.4. Functions of one or two variables

When the h approximation contains only functions of one variable (20), the constraints in (ALP) with the truncation $x_i < N$ have the form

$$\sum_{i=1}^{n} f_i(x_i, u_{(i)}) \geq 0 \quad \text{for all } 0 \leq x_i < N,\ u \in \mathcal{A}(x) \tag{34}$$

where $u_{(i)}$ is a vector containing some of the components of u. For example, if each class has at most one predecessor, (23) implies that $u_{(i)}$ contains u_i and, if class i has a predecessor, $u_{p(i)}$. Introduce the variables $f_{i+n}(u_{(i)})$, representing the minimum of $f_i(x_i, u_{(i)})$ over x_i. Then (34) is equivalent to

$$\sum_{i=1}^{n} f_{i+n}(u_{(i)}) \geq 0 \text{ for all } u \in \mathcal{A} \tag{35}$$

$$f_{i+n}(u_{(i)}) \leq f_i(x_i, u_{(i)}) \quad \text{for all } x_i < N \text{ such that } u_{(i)} \text{ is feasible}, \quad i = 1, \ldots, n. \tag{36}$$

Constraint (36) is needed for those $u_{(i)}$ appearing in (35), namely, those $u_{(i)}$ that can be augmented to give a feasible action $u \in \mathcal{A}$. The constraint reduction is considerable. For a series line, $u_{(1)}$ and $u_{(n)}$ have two feasible values and the other $u_{(i)}$ have four feasible values (two components), giving less than $(4n-4)N$ constraints (36) and 2^{n-1} constraints (35). For comparison, (34) contains nearly $2^{n-1}n^N$ constraints.

Further reduction is achieved by eliminating certain u_i. Note that (36) eliminates x_i because it only appears in f_i. If a variable u_j appears in $u_{(i)}$ but does not appear with any other variables x_k, $k \neq i$, then u_j can be eliminated in the same manner as x_i. Indeed, this elimination can be done sequentially, replacing f_i one at a time by f_{i+n} and considering only the remaining variables. For example, the constraints

$$f_1(x_1, u_1) + f_2(x_2, u_1, u_2) + f_3(x_3, u_2, u_3) \geq 0,\ u_i \leq x_i < N,\ u_i = 0, 1$$

are equivalent to

$$f_6(u_2) \leq f_3(x_3, u_2, u_3) \quad \text{for all } x_3, u_2, u_3 \tag{37}$$

and

$$f_1(x_1, u_1) + f_2(x_2, u_1, u_2) + f_6(u_2) \geq 0 \quad \text{for all } x_1, x_2, u_1, u_2.$$

Now that u_2 does not appear with x_3, eliminate x_2 and u_2, giving the equivalent constraints (37),

$$f_5(u_1) \leq f_2(x_2, u_1, u_2) + f_6(u_2) \quad \text{for all } x_2, u_1, u_2$$

and

$$f_1(x_1, u_1) + f_5(u_1) \geq 0 \quad \text{for all } x_1, u_1.$$

There are $6N - 3$ reduced constraints.

Note that the feasible u in this example is for a single-class network. If the network is multiclass, a modified approach is needed in order to determine the feasible actions in constraints such as (37). At some point in the elimination process, suppose u_j appears in $u_{(i)}$ but does not appear with any other variables x_k, $k \neq i$. Before eliminating u_j, augment $u_{(i)}$ with all other classes k

that use the same server, $\sigma(k)=\sigma(j)$, and appear in the constraint after f_i is eliminated.

This method can be extended to functions of two or more variables using ideas from factored MDPs. An MDP on a state space X is factored if

$$X = X_1 \times X_2 \times \cdots \times X_s$$

$$p_u(x,y) = \prod_{j=1}^{s} p^{(j)}_{u_{(j)}}(x_{(j)}, y_{(j)})$$

and cost is additive over $x_{(j)}$. Here X_j contains a subset of "local" state variables of X and $x_{(j)}$, $u_{(j)}$ are vectors containing some of the components of x and u. The factored assumption allows a more compact representation of the MDP, but it is no easier to solve because the differential cost needed in (1) does not factor. Turning to ALP, if each ϕ_k is a function of some $x_{(j)}$, then h is also "factored" and the constraints in (ALP) can be written as a sum of functions of the factored state variables and actions

$$\sum_{j=1}^{s} f_j(x_{(j)}, u_{(j)}) \geq 0 \quad \text{for all } x \in X,\ u \in \mathcal{A}(x). \tag{38}$$

Returning to the MQNET problem, no factored representation of the transition probabilities appears possible, particularly after uniformization. However, computing the expectation $(P_u h)(x)$ is not difficult and it suffices that the cost $c'x$ and the h approximation "factor". As in the example above, one can sequentially eliminate f_j's and their unique variables. For brevity, we only describe functions of overlapping pairs of variables, $x_{(j)} = (x_j, x_{j+1})$, $j = 1, \ldots, n-1$. Eliminating just the x variables, (38) is equivalent to

$$\sum_{i=1}^{n-1} f_{i+n}(u_i, u_{i+1}) \geq 0 \quad \text{for all } u \in \mathcal{A} \tag{39}$$

$$f_{i+n}(u_{(i)}) \leq f_i(x_i, x_{i+1}, u_{(i)}) \tag{40}$$

for all $u_i \leq x_i < N$, $u_{i+1} \leq x_{i+1} < N$, and $u_{(i)}$ feasible for $i = 1, \ldots, n-1$. As described above for functions of one variable, certain u_j can also be eliminated after augmenting $u_{(i)}$ with other classes that use the same servers. There are roughly N^2 constraints in (40) for each $u_{(i)}$ and $|\mathcal{A}|$ constraints in (39). For the series line, there are at most 8 feasible $u_{(i)}$ (three components).

6. Numerical results

ALPs were solved for a variety of networks with up to 11 classes and series lines with up to 17 classes. For the smaller examples, average cost error was computed by also solving the DP. The numerical methods and examples are described in Section 6.1. Results for the smaller approximation architectures are given in Section 6.2 and for larger approximation architectures in Section 6.3. Performance of the ALP policy is covered in Section 6.4.

6.1. Methods and examples

The ALPs were solved using QNET Approximator, available at www.math-cs.gordon.edu/qna, and CPLEX, except that the arrival routing problem was solved using a specialized program. Constraint reduction was only implemented for quadratic functions; all other ALPs use the constraint sampling and limiting constraint method described in Appendix B. Sample sizes were set by increasing sample size until the change in average cost was small. The DPs were solved using value iteration on a truncated state space, stopping at an error tolerance of 0.001. Most of the DP results use the program at www.math-cs.gordon.edu/~senning/qnetdp. Truncations for the DP were set by checking sensitivity, with an accuracy goal of 0.1%. For the larger examples, distinct truncations were set for each variable x_i.

ALP run times for quadratic functions and functions of one variable were quite fast. A 17 class series line with quadratic approximation was solved in 132 s using CPLEX version 10 on a 64-bit, 3.4 GHz processor with 6 GB of RAM, including generating and solving the LP. All other run times are for CPLEX version 12 on a 1.86 GHz dual-core processor with 6 GB of RAM and do not include generating the LP. For functions of one variable, the 11-class network with 199 variables was solved in 238 s. In comparison, the DP for the four-class series queue took 96 s for the smallest run with an error less than 15%, 352 s for an error less than 5%, and 1168 s for an error less than 1%. It was run using a parallel implementation on a cluster of eight workstations; the reported time is clock time multiplied by eight. Generally, the ALP is much faster at low levels of accuracy but loses its advantage as the accuracy requirement increases. The main reason is that the LP solution time increases more rapidly with the number of variables than does DP.

Examples include MQNETs (as modeled in Section 2), arrival routing, and a system with a cross-trained server. The ALP and quadratic constraint reduction are easily modified for these other networks. Parameter values are shown in Table 4. Series queues with more than four classes are also used; like the four-class example, $\mu = (1+0.1(n-1), \ldots 1)$ and $c = (1, \ldots, 2)$, with the μ_i and c_i equally spaced. In the parallel server "N" network from [36], there are two classes and two servers. Server 1 can only serve class 1. Server 2 can serve class 1 or class 2. In the arrival routing problem from [37], arrivals must be immediately routed to one of two classes, each with its own server. The reentrant line of Fig. 1 is studied in [38,34]. The Rybko–Stoylar network, shown in Fig. 2, is considered a challenging example because some static priority policies are not stabilizing. The six-class, two-station network of Fig. 3 is a modification of [39] studied in [40].

An 11-class, four-station network with reentrant flow and rework, typical in manufacturing applications, is taken from [31, Figure 7.1 and

Table 4
Parameters of the examples.

	α	μ	c	$\rho_{\max}$
Series queue ($\mu_1 > \mu_2$)	1	1.5, 1.25	1,2	0.8
Series queue ($\mu_1 < \mu_2$)	1	1.15, 1.4	1,3	0.87
Series queue ($\mu_1 = \mu_2$)	varies	1, 1	1, 2	–
2-Class "N"	1.2, 0.4	1, 1, 1	1, 1	0.8
Arrival routing	1	0.65, 0.65	1, 2	0.77
3-Class reentrant line	0.1429	0.6, 0.16, 0.25	1,1,1	0.89
4-Class series queue	0.8	1.3, 1.2, 1.1, 1.0	1, 4/3, 5/3, 2	0.8
Rybko–Stolyar	.0672, .0672	0.12, 0.12, 0.28, 0.28	1, …, 1	0.8
6-Class network	6/140, 6/140	1/4, 1, 1/8, 1/6, 1/2, 1/7	1, …, 1	0.6

Section 7.2.1] and adapted to our problem setting. Station 1 serves classes 1–4, station 2 serves 5–9, station 3 serves 10, and station 4 serves 11. Station 5 in [31], which serves one class and has low traffic intensity, is omitted. Job type 1 follows a route through classes 1, 5, 9, and 4. Job type 2 is routed 2, 6, 10, 11 or 8, 3, 7. The routing from class 10 is probabilistic, with $p_{10,8} = 1 - p_{10,11} = 3/19$. In class 11 or 8, class 3 is visited next. There is also probabilistic routing from class 7 to class 10, with $p_{7,10} = 0.2$ representing rework. The other parameters are $\alpha_1 = \alpha_2 = 19$, $\mu = (65, 130, 65, 130, 75, 150, 75, 150, 75, 25, 37.5)$, $c = 1$, and $\rho = (0.95, 0.975, 0.95, 0.533)$.

6.2. Quadratics and functions of one variable

Accuracy of the average cost bound for quadratic approximation and for functions of one queue length is shown in Table 5. The quadratic approximation is not very accurate, particularly for the series queues. However, Section 6.4 shows that accuracy improves in light traffic or single-bottleneck heavy traffic. Adding functions of one queue length, (21) and (22), improves accuracy to 12–30%. For the 11-class network, optimal average cost is not available, but it appears that functions of one variable are not very effective. We used $M = 10$ functions ϕ_j and $\phi_0(x_i) = x_i \ln x_i$ for each class and parameters $\gamma = 0.05$ and $B = 0.95$ in (22). An LP size of 13×6 indicates 13 constraints and 5 basis functions plus the variable J. The ALPs with functions of one variable contain a large number of constraints because constraint sampling was used.

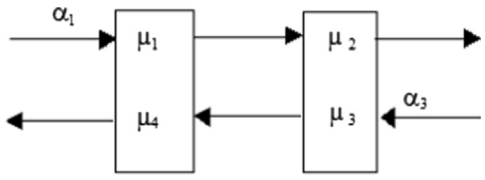

Fig. 2. The Rybko–Stolyar network.

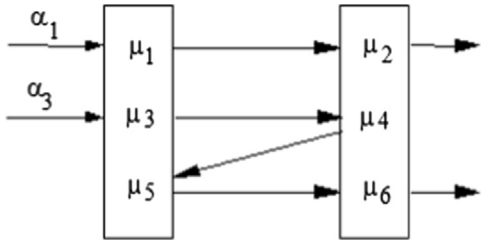

Fig. 3. A six-class network.

Table 5
Accuracy of the ALP average cost.

Network	Optimal average cost	Error in ALP J (size of LP)	
		Quadratic	Quadratic + one var
Series queue ($\mu_1 > \mu_2$)	9.31	−40% (13 × 6)	−13% (300 K × 28)
Series queue ($\mu_1 < \mu_2$)	13.50	−27% (13 × 6)	−14% (300 K × 28)
2-Class "N"	4.54	−12.0% (11 × 6)	−11.9% (60 K × 28)
Arrival routing	5.54	−19%(13 × 6)	–
3-class reentrant line	11.47	−19% (18 × 10)	−17% (209 K × 43)
4-Class series queue	16.30	−52% (80 × 15)	−30% (100 K × 59)
Rybko– Stolyar	6.87	−32% (45 × 15)	−28% (200 K × 59)
6-Class network	2.56	−19% (89 × 28)	−15% (200 K × 94)
11-Class network	–	31.71[a] (1440 × 78)	31.88[a] (446 K × 199)

[a] Average cost.

6.3. Larger approximation architectures

Extensive tests were conducted using the additional approximating functions in Section 4 to investigate the trade-off between size and accuracy using ALP. Results for three progressively larger examples are shown in Figs. 4–6. Of the many approximations tested, only those with good accuracy for their size are graphed. Piece-wise quadratics on cones were not included in the tests. For comparison, a simple "indicator" approximation is shown that uses quadratic functions and indicator functions on individual states with total queue length up to some M, i.e., in states x such that $\sum_{i=1}^{n} x_i \le M$. The optimality equation is satisfied in these states. The accuracy of using DP on a truncated state space, $x_i < N$ for some buffer limit N, is also shown. The DP has one variable for each state, so the horizontal axis compares the number of variables.

For these examples, the best ALPs shown achieve the same accuracy as the truncated DP with roughly two orders of magnitude fewer variables. The ALP does not do quite as well in the six-class example, where the smaller traffic intensity of 0.6 makes the problem easier for DP. The ALPs have the greatest advantage when the accuracy requirement is modest, such as 10 or 20% error. For the four and six-class examples, ALPs with error of less than 4 or 5% could not be solved with the available memory, i.e., results were still sensitive to sample size for the largest sample size that could be run.

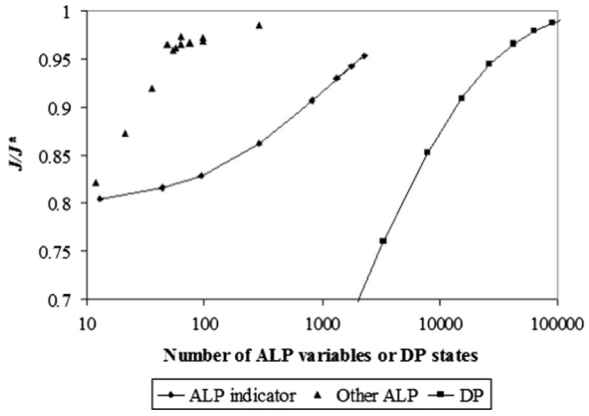

Fig. 4. Accuracy and size of ALPs, three-class reentrant line.

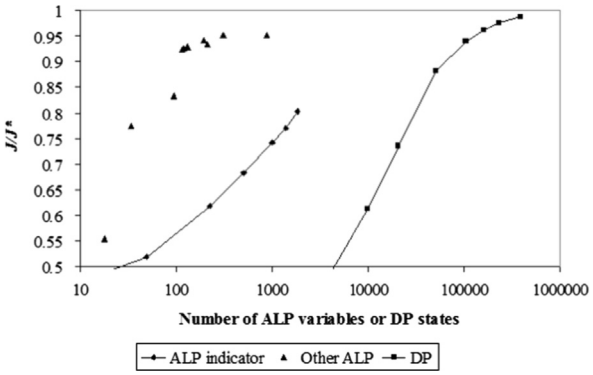

Fig. 5. Accuracy and size of ALPs, four-class series queue.

Table 6 lists the approximation architectures with the best size-accuracy trade-off. The parameters listed are

- β_i, the base for the exponential functions (13) and (14).
- N_i, the maximum queue length for the rational functions (17).
- M, which controls the number of functions of one or two variables (18) or piece-wise quadratic and linear (21).
- γ, B for the functions of one or two variables (22).

The functions of one variable are functions of queue length and the functions of two variables, or "pairs", are functions of queue length and echelon inventory. For the six-class network, functions of workload also improved accuracy and are used in one run in Fig. 6. Adding exponential functions and functions of one variable increased the average cost bound 6% compared to just quadratics for the 11-class network. These results suggest an order for adding functions to the approximation architecture: (i) quadratics, (ii) exponential functions, (iii) rational functions (for smaller networks), (iv) functions of one or two variables, and (v) piece-wise linear or quadratic (PWL/PWQ) on rectangular regions.

6.4. Performance

Performance of the h-greedy policy for the ALP solution h, found using value iteration for this fixed policy, was tested for the two-class series queue. The quadratic and exponential approximations (A.1), called EXP3, were used. In Fig. 7, traffic intensity $\rho=\alpha/\mu_2$ was varied while keeping μ_i fixed at the values for the $\mu_1>\mu_2$ example in Section 6.1. The data below the line at 1.0 shows the accuracy of the lower bound, i.e., $\underline{J}/J_*$, and the data above 1.0 shows the performance, i.e., J_u/J_*. The percent error vanishes in light traffic. The data suggests that percent error also vanishes in heavy traffic. This is not surprising, since there is a single bottleneck at station 2. The quadratic ALP is known to give a tighter bound than the achievable region method, which is known to have a vanishing percent error in heavy traffic. Performance of the ALP policy is within 10% of optimal 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 (A.1) sometimes has poorer performance than the quadratic, even though it includes quadratic functions.

The quadratic ALP policy for this example has a simple form: server 1 is busy when $x_2\le 1$ and $x_2>0$. Using a quadratic and the function β^{x_2}, where $\beta=\mu_2/\mu_1=0.8333$, the policy for this approximation at $\alpha=1$ is shown in Fig. 8. Both the ALP and optimal policy have switching curve form: for a given x_1, server 1 idles when x_2 is below the switching curve. The ALP switching curve is quite close to the optimal switching curve and has an average cost just 1% above optimal. However, such close fits do not appear to be the norm; using (A.1), which has two more exponential terms, has a nearly flat switching curve at $x_2=5$, which does not track the optimal switching curve as closely, and an average cost 9% above optimal.

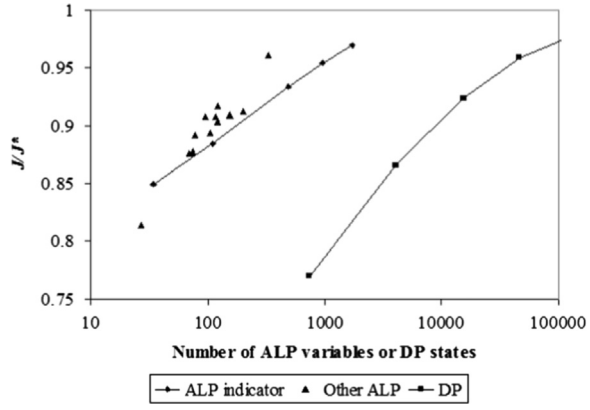

Fig. 6. Accuracy and size of ALPs, six-class network.

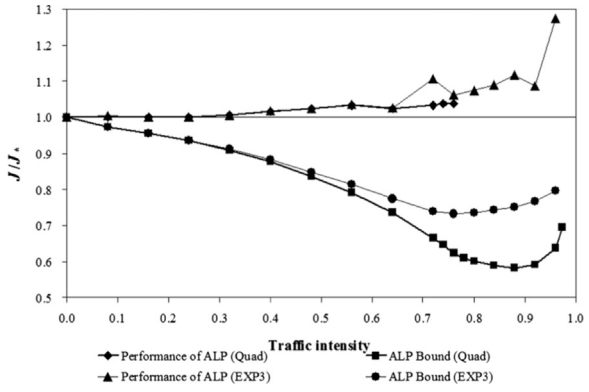

Fig. 7. Performance of the ALP policy, two-station series queue, $\mu_1>\mu_2$.

Table 6
Percent error and solution time for the best ALPs.

Approximation	β	N	M	γ	B	Error	Var's	Time (s)
3-Class reentrant line								
Quad+exp	0.9, 0.27, 0.9					−12.7	22	20
Quad+exp+rat'l	0.9, 0.27, 0.9	31, 56, 31				−5.6	49	47
Quad+exp+rat'l+one var	0.9, 0.27, 0.9	31, 56, 31	5	0	0.85	−2.7	64	32
Exp+rat'l+PWQ	0.9, 0.27, 0.9	31, 56, 31	4			−1.4	295	93
4-Class series queue								
Quad+exp	0.9	15, 18, 21, 30				−22.6	35	23
Quad+exp+rat'l	0.9	15, 18, 21, 30				−7.6	116	67
Quad+exp+rat'l+pair	0.9	15, 18, 21, 30	5			−5.9	195	89
Quad+exp+rat'l+PWL	0.9	15, 18, 21, 30	3			−4.8	308	180
6-Class network								
Quad+exp+pair	0.5		3	0.05	0.95	−8.3	121	26
Quad+exp+PWL	0.5		2			−3.9	331	242
11-Class network								
Quad+exp+one var	0.5		5	0	1	33.64[a]	276	681

[a] Average cost.

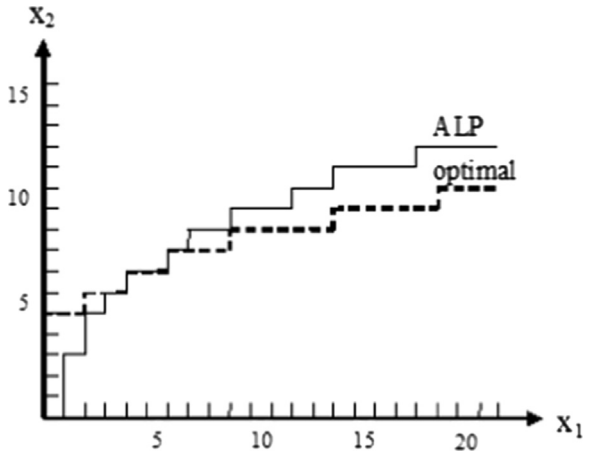

Fig. 8. Policy for the ALP with quadratic and β^{x_2} functions, series line ($\mu_1 > \mu_2$). Server 1 idles below the switching curve.

The performance of both ALP policies is better, with error under 5%, for the series queue with $\mu_1 < \mu_2$ and with $\mu_1 = \mu_2$. Traffic intensity affects error in a similar way when the piecewise quadratic approximation (16) is used for the three-class reentrant line, again with a single bottleneck station. However, for the series line with $\mu_1 = \mu_2$ which has two bottlenecks, as traffic intensity increases, the accuracy of the ALP bound continues to degrade.

7. Conclusion

We have demonstrated the feasibility of using ALP to compute useful lower bounds on optimal average cost for moderate size networks:

- The bounds are tighter than previous bounds, surveyed in [41]. Even the simplest, quadratic ALP gives tighter bounds than the achievable region LP in [3,42]; see [43]. This ALP was solved quickly for up to 17 buffers.
- Accuracy of 1–5% was achieved for networks with up to six buffers by adding other approximating functions. They are significantly more accurate than any functions previously proposed, but at the cost of using more functions. An order for systematically adding functions to improve accuracy was developed.
- The best ALPs require much less computation than DP value iteration to achieve a given accuracy; however, this advantage diminishes when more accuracy is desired.
- Constraint reduction techniques improve the tractability of ALPs with certain functions.
- Constraint sampling did not perform as well as expected: a very large number of constraints were required. A hybrid constraint sampling and truncation method allows a smaller number of sampled constraints to be used.
- A useful policy can also be obtained for some, but not all, networks.

Some tuning of the approximation architecture was done to achieve the most accuracy in examples, and further tuning could be done. However, the set of functions and order for adding the functions appears fairly robust, i.e., it gives an average cost bound with good accuracy on this class of problems. Some of the functions are likely to be useful for value function approximation in other problem settings. A major question is how to automate the choice of functions. Work on this problem in the context of reinforcement learning has used adaptive state aggregation and local approximating functions; see [44]. Their method attempts to automate what we did empirically, in examining Bellman error to construct new functions. Non-parametric ADP methods have also been developed recently that avoid choosing approximating functions. For example, [5] uses constraint sampling and solves the dual of a nonparametric form of the smoothed ALP in [7]. They report performance for the Rybko–Stolyar example described in Section 6 with discounted cost. It would be interesting to develop such a method for average cost bounds.

Much more could be done to solve the ALP efficiently. In [14], ALPs with quadratic approximation and up to 40 buffers are solved using column generation on the dual. An open question is whether column selection can be done efficiently for other approximation architectures. The number of constraints could also be reduced by implementing the restriction to nonidling policies (Section 5.1), heuristic policy restrictions, and the constraint reduction methods of Sections 5.2–5.4. When constraint sampling is needed, better distributions could be obtained, e.g., by simulating a heuristic policy. Average cost ALPs are very sensitive to limiting constraints; a Lagrangian relaxation approach as in [7] might address this issue. Using customized LP algorithms as in [7] could also be valuable. It should be noted, however, that the ALPs (other than the quadratic approximation) do not scale to very large networks because of the number of functions. In that setting, methods that decompose the problem into smaller networks, simulation-based methods, or ALPs with quadratic approximations are more appropriate and one should expect looser bounds or policies.

Acknowledgments

The numerical work in this paper was done by my colleague Jonathan Senning and our students Taylor Carr, Adam Elnagger, Christopher Pfohl, Lauren Berger, Lauren Carter, Jane Eisenhauer, Jeff Fraser, Michael Frechette, Melissa LeClair, Lauren Meitzler, Josh Nasman, and Nathan Walker. I would also like to thank Sean Meyn for his many suggestions. This work was supported in part by National Science Foundation Grant 0620787.

Appendix A. ALP constraints for the series queue

This appendix derives and reduces the ALP constraints for the series queue and the differential cost approximation

$$h(x) = \tfrac{1}{2}x'Qx + px + r_1\beta_2^{x_2} + r_2x_1\beta_2^{x_2} + r_3x_2\beta_2^{x_2}. \tag{A.1}$$

Assume $c_1 < c_2$, so that only station 2 is nonidling, and $\mu_2 \le \mu_1$. Set $\beta \equiv \beta_2 = (\mu_2/\mu_1)$, $\alpha \equiv \alpha_1$ (the arrival rate), and recall that $x = z + u$.

For this problem, (23) is

$$J \le c'x + \alpha\left[h(x+e_1) - h(x)\right] + u_1\mu_1\left[h(x-e_1+e_2) - h(x)\right] + u_2\mu_2\left[h(x-e_2) - h(x)\right]. \tag{A.2}$$

Substituting (A.1) into (A.2), (A.2) has the form

$$J \le d^u + (c^u)'z + (\zeta^u + (\xi^u)'z)\beta^{z_2+u_2} \tag{A.3}$$

for all $z \in Z_+^2$ and all u that are nonidling at station 2. Next we express d^u, c^u, ζ^u and ξ^u as linear functions of the variables p, Q, and r.

The terms in (A.2) are

$$h(x+e_1) - h(x) = q_{11}x_1 + q_{12}x_2 + \frac{1}{2}q_{11} + p_1 + r_2\beta^{x_2}$$

$$h(x-e_1+e_2) - h(x) = (-q_{11}+q_{12})x_1 + (q_{22}-q_{12})x_2 + \frac{1}{2}q_{11} + \frac{1}{2}q_{22} - q_{12} - p_1 + p_2 - r_1(1-\beta)\beta^{x_2} - r_2[(1-\beta)x_1 + \beta]\beta^{x_2} - r_3[(1-\beta)x_2 - \beta]\beta^{x_2}$$

$$h(x-e_2)-h(x)=-q_{12}x_1-q_{22}x_2+\frac{1}{2}q_{22}-p_2+r_1\left(\frac{\mu_1}{\mu_2}-1\right)\beta^{x_2}$$
$$+r_2x_1\left(\frac{\mu_1}{\mu_2}-1\right)\beta^{x_2}+r_3\left[\left(\frac{\mu_1}{\mu_2}-1\right)x_2-\frac{\mu_1}{\mu_2}\right]\beta^{x_2}.$$

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

$$c_1^{(1,1)}=c_1-(\mu_1-\alpha)q_{11}+(\mu_1-\mu_2)q_{12}$$
$$c_2^{(1,1)}=c_2-(\mu_1-\alpha)q_{12}+(\mu_1-\mu_2)q_{22}$$
$$d^{(1,1)}=c_1^{(1,1)}+c_2^{(1,1)}+\frac{1}{2}(\alpha+\mu_1)q_{11}-\mu_1q_{12}+\frac{1}{2}(\mu_1+\mu_2)q_{22}$$
$$-(\mu_1-\alpha)p_1+(\mu_1-\mu_2)p_2$$
$$\zeta^{(1,1)}=-r_2(\mu_2-\alpha)-r_3(\mu_1-\mu_2).$$

For $u=(0,1)$,

$$c_1^{(0,1)}=c_1+\alpha q_{11}-\mu_2q_{12}$$
$$c_2^{(0,1)}=c_2+\alpha q_{12}-\mu_2q_{22}$$
$$d^{(0,1)}=c_2^{(0,1)}+\tfrac{1}{2}\alpha q_{11}+\tfrac{1}{2}\mu_2q_{22}+\alpha p_1-\mu_2p_2$$
$$\xi_1^{(0,1)}=r_2(\mu_1-\mu_2)$$
$$\xi_2^{(0,1)}=r_3(\mu_1-\mu_2)$$
$$\zeta^{(0,1)}=\xi_2^{(0,1)}+r_1(\mu_1-\mu_2)+r_2\alpha-r_3\mu_1.$$

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

$$c_1^{(1,0)}=c_1-(\mu_1-\alpha)q_{11}+\mu_1q_{12}-r_2(\mu_1-\mu_2)$$
$$d^{(1,0)}=c_1^{(1,0)}+\tfrac{1}{2}(\alpha+\mu_1)q_{11}-\mu_1q_{12}+\tfrac{1}{2}\mu_1q_{22}-(\mu_1-\alpha)p_1+\mu_1p_2$$
$$-r_1(\mu_1-\mu_2)-r_2(\mu_2-\alpha)+r_3\mu_2.$$

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

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

Now we reduce the constraints (A.3). For $u=(1,1)$, $\xi^{(1,1)}=0$, so as $z_i\to\infty$ we must have

$$c_i^{(1,1)}\geq 0,\quad i=1,2. \tag{A.4}$$

Given (A.4), (A.3) 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 (A.3) at $z_1=0$, $z_2=0,\ldots,N-1$ 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

$$c_1^{(0,1)}+\beta\xi_1^{(0,1)}\geq 0 \tag{A.5}$$

and $c_1^{(0,1)}\geq 0$. Letting $z_2\to\infty$ in (A.3) gives another constraint, so we have

$$c_i^{(0,1)}\geq 0,\quad i=1,2. \tag{A.6}$$

Given (A.5) and (A.6), (A.3) 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 (A.3) at $u=(0,1)$, $z_1=0$, and $z_2=0,\ldots,N-1$. Next, for $u=(1,0)$ we must have $z_2=0$. For (A.3) to hold as $z_1\to\infty$, we must have

$$c_1^{(1,0)}\geq 0. \tag{A.7}$$

In light of (A.7), (A.3) is tightest at $z_1=0$, so we include (A.3) at $u=(1,0)$ and $z=(0,0)$. Finally, we include (A.3) at $u=z=(0,0)$.

To summarize, the approximate reduced ALP contains the $2N+8$ constraints (A.3) at $u=(1,1), z_1=0$, $z_2=0,\ldots,N-1$; $u=(0,1), z_1=0$, $z_2=0,\ldots,N-1$; $u=(1,0), z=(0,0)$; and $u=z=0$; plus (A.4)–(A.7).

Appendix B. Constraint sampling

In constraint sampling, only a weighted sample of the constraints in (ALP) is used. We sampled states according to the probability distribution

$$\pi(x)=\prod_{i=1}^{n}(1-\overline{\rho}_i)\overline{\rho}_i^{x_i}.$$

However, because it may be difficult to generate enough unique states, $1-\overline{\rho}_i$ is repeatedly reduced by a constant factor during sampling to spread the distribution. For most runs, the initial value of $\overline{\rho}_i$ is the traffic intensity of station $\sigma(i)$. Where noted, the value 0.5 was used because it performed better. When a state x is sampled, constraints for all actions $u\in\mathcal{A}(x)$ are generated. Also, constraints for the 2^n states with $x_i=0$ or 1 are generated before sampling because of their potential importance.

Table B1 shows that even for the two-class series queue, the sample size required for a small average cost error due to sampling depends significantly on the approximation architecture. Exponential functions (13) and (14) with $\beta=(0.9,0.8333)$, the rational functions (17) with $N_i=12$, and functions of one queue length (21) with $M=10$ and no ϕ_0 are used. The sampling distribution was

Table B1
Accuracy of constraint sampling for the series queue ($\mu_1>\mu_2$). Reported error is the mean of five runs compared to the ALP without sampling.

Constraints approximation	Sampling error, %			
	200	500	1000	2000
Quadratic	0	0	0	0
Quad+exp	10.8	3.4	0.1	
Quad+rat'l	2.5	0.8		
Quad+exp+rat'l	1.5	0.6		
Quad+one var	7.6	3.0	1.4	0.6

Table B2
Accuracy of constraint sampling with limiting constraints, series queue. Error is the mean of five runs compared to the most accurate run available.

Constraints approximation	Sampling error (with limiting constraints), %						
	5000	10,000	50,000	100,000	500,000	10^6	2×10^6
4-Class series queue							
Quadratic	1.2(0.0)	0.9	0.5	0.4	0.26		
Quad+exp				5.9(3.9)	2.6(1.9)	1.6(1.1)	0.6(0.3)
Quad+exp+rat'l		2.2(1.0)	1.1(0.6)	0.7(0.5)	0.28(0.19)	0.21[a](0.08)	
Quad+one var	1.6(0.09)	0.09(0.09)					
8-Class series queue							
Quadratic	(0.0)			13.4	6.5	4.7	2.8[a]
Quad+exp			(8.8)	22.9(5.5)	9.7(0.8)	8.9(0.0)	
Quad+one var				15.7(5.7)	3.7(0.9)	1.6(0.0)	

[a] Mean of two runs.

initialized with $\overline{\rho}_i = 0.5$. There is no sampling error with quadratic approximation; the $x_i = 0$ or 1 constraints and a few randomly sampled constraints are sufficient.

Table B2 shows sampling error for two larger examples. For both examples, $\beta_i = 0.9$ and $N_i = 12$. For the four-class series queue, $M = 10$ and $\overline{\rho}_i = 0.5$. For the eight-class series queue, the data are $\alpha = 1$, $\mu = (1.75, \ldots, 1.25)$, and $c = (1, \ldots, 2)$, with the μ_i and c_i equally spaced, and the parameters are $M = 5$ and $\overline{\rho}_i = \alpha/\mu_i$. Again the sample size required is highly dependent on the approximation architecture. Exponential functions require two million constraints for a sampling error of less than 1% in the four-class series queue. Actual errors are somewhat larger than reported for the non-quadratic cases because error is calculated in comparison to the best available run (the smallest average cost). For example, the four-class quadratic plus exponential row is in comparison to the minimum of 10 runs made with two million constraints. In Tables B1 and B2, the coefficient of variation of the error is small, generally less than 0.2, so that the mean of five trials is a reasonable estimate.

To reduce sampling error, we used a hybrid sampling and truncation method. The rationale for this method is that constraints tend to vary slowly and consistently with x for a given u. Sampling and truncation both tend to miss the effect of constraints at large x. The *limiting constraints* results in Table B2 add constraints, for each u, at the $2^n - 1$ states $x \neq u$ such that $x_i = u_i$ or $\overline{N}$. The idea is that for large $\overline{N}$, these constraints approximate a limiting constraint in various directions in the state space. To avoid numerical issues, $\overline{N} = 500$ was used (10,000 for quadratic functions and eight-class with quadratic plus one variable). The limiting constraints were not included in the number of constraints reported. Table B2 shows that limiting constraints can reduce the sample size required by an order of magnitude or more. The limiting constraints virtually eliminate sampling error for the quadratic approximation because they emulate (25).

The results in Section 6 use truncation for examples with two or three classes and constraint sampling for larger examples. The ALP truncations were set by checking the sensitivity of average cost to the truncation in an effort to keep truncation error less than 1–2% and similarly for sample size. Sensitivity was checked for each network and many approximation architectures; however, once truncations or sample size were set, they were used for similar approximation architectures. Limiting constraints at $\overline{N} = 500$ were used for the three-class reentrant line, four-class series queue, Rybko–Stolyar, six-class, and 11-class examples in Section 6.2. For the two-class series queue, the truncation (400, 400) was used, i.e., the constraints for $x_1 \leq 400$ and $x_2 \leq 400$. For the "N" network, the truncation was (100, 100) and for the three-class reentrant line (40, 80, 20). The following number of constraints was used in Section 6.2. For the four-class series queue, 100,000 constraints were sampled; 200,000 were sampled for the Rybko–Stolyar, six-class, and 11-class networks. In all but the 11-class network, the number of limiting constraints is relatively small and they are not included in the constraint count. For the 11-class network, there were 200,000 sampled constraints plus 246,000 limiting constraints; the $x_i = 0$ or 1 constraints were too numerous to include. In Section 6.3, where more approximating functions are used, the number of constraints sampled was increased to 200,000 for the four-class series queue but reduced to 100,000 for the six-class network to reduce run times.

References

[1] Papadimitriou C, Tsitsiklis J. The complexity of optimal queuing network control. Math Oper Res 1999;24:293–305.

[2] Bertsimas D, Gamarnik D, Tsitsiklis J. Performance of multiclass Markovian queueing networks via piecewise linear Lyapunov functions. Ann Appl Probab 2001;11:1384–428.

[3] Bertsimas D, Paschaladis I, Tsitsiklis J. Optimization of multiclass queueing networks: polyhedral and nonlinear characterizations of achievable performance. Ann Appl Probab 1994;4:43–75.

[4] de Farias D, Roy BV. Approximate linear programming for average-cost dynamic programming. In: Advances in neural information processing systems, vol. 15. Cambridge, MA: MIT Press; 2003.

[5] Bhat N, Farias V, Moallemi C. Non-parametric approximate dynamic programming via the kernel method. Working paper. Columbia University; 2012.

[6] Moallemi C, Kumar S, Roy BV. Approximate and data-driven dynamic programming for queueing networks. Working paper. Columbia University; 2008.

[7] Desai V, Farias V, Moallemi C. Approximate dynamic programming via a smoothed linear program. Oper Res 2012;60(3):655–74.

[8] Powell W. Approximate dynamic programming: solving the curses of dimensionality. Hoboken, NJ: Wiley; 2007.

[9] de Farias D, Roy BV. On constraint sampling for the linear programming approach to approximate dynamic programming. Math Oper Res 2004;29(3):462–78.

[10] Schweitzer P, Seidmann A. Generalized polynomial approximations in Markovian decision processes. J Math Anal Appl 1985;110:568–82.

[11] de Farias D, Roy BV. The linear programming approach to approximate dynamic programming. Oper Res 2003;51(6):850–65.

[12] de Farias D, Roy BV. A linear program for Bellman error minimization with performance guarantees. In: Advances in neural information processing systems, vol. 17. Cambridge, MA: MIT Press; 2005.

[13] Veatch M. Approximate linear programming for average cost MDPs. Math Oper Res 2013;38(3):535–44.

[14] Veatch M, Walker N. Approximate linear programming for network control: Column generation and subproblems. Working paper. Gordon College, Department of Mathematics. Available at ⟨http://faculty.gordon.edu/ns/mc/mike-veatch⟩; 2008.

[15] Morrison J, Kumar P. New linear program performance bounds for queueing networks. J Optim Theory Appl 1999;100(3):575–97.

[16] Morrison J, Kumar P. Computational performance bounds for Markov chains with applications. IEEE Trans Autom Control 2008;53:1306–11 Full-length version available at ⟨http://black.csl.uiuc.edu/~prkumar⟨.

[17] Koller D, Parr R. Computing factored value functions for policies in structured MDPs. In: IJCAI'99: proceedings of the 16th international joint conference on artificial intelligence. San Francisco: Morgan Kaufmann Publishers; 1999. p. 1332–9.

[18] Guestrin C, Koller D, Parr R. Max-norm projections for factored MDPs. In: IJCAI'01: proceedings of the 17th international joint conference on artificial intelligence, vol. 1. San Francisco: Morgan Kaufmann Publishers Inc.; 2001. p. 673–80.

[19] Schuurmans D, Patrascu R. Direct value-approximation for factored MDPs. Adv Neural Inf Process Syst 2001;14:1579–86.

[20] Guestrin C, Koller D, Parr R, Venkataraman S. Efficient solution algorithms for factored MDPs. J Artif Intell Res 2003;19:399–468.

[21] Bertsekas D. Dynamic programming and optimal control, 3rd ed., vol. 2. Belmont, MA: Athena Scientific; 2007.

[22] Rogers L. Pathwise stochastic optimal control. SIAM J Control Optim 2007;46(3):1116–32.

[23] Brown D, Smith J, Sun P. Information relaxations and duality in stochastic dynamic programs. Oper Res 2010;58(4-Part-1):785–801.

[24] Desai V, Farias V, Moallemi C. Pathwise optimization for optimal stopping problems. Manag Sci 2012;58(12):2292–308.

[25] Desai V, Farias V, Moallemi C. Bounds for Markov decision processes. In: Reinforcement learning and approximate dynamic programming for feedback control; 2011. p. 452–73.

[26] Sennott L. Stochastic dynamic programming and the control of queueing systems. New York: Wiley; 1999.

[27] Meyn S. Sequencing and routing in multiclass queueing networks. Part I: feedback regulation. SIAM J Control Optim 2001;40:741–76.

[28] Meyn S. The policy iteration algorithm for Markov decision processes with general state space. IEEE Trans Automat Control 1997;AC-42:191–7.

[29] Meyn S. Stability, performance evaluation and optimization. In: Feinberg E, Shwartz A, editors. Handbook of Markov decision processes: methods and applications. Boston, MA: Kluwer; 2001.

[30] Veatch M. Using fluid solutions in dynamic scheduling. In: Gershwin SB, Dallery Y, Papadopoulos CT, Smith JM, editors. Analysis and modeling of manufacturing systems. New York: Kluwer; 2002. p. 399–426.

[31] Meyn S. Control techniques for complex networks. New York: Cambridge University Press; 2008.

[32] Meyn S. Dynamic safety-stocks for asymptotic optimality in stochastic networks. Queueing Syst Theory Appl 2005;50:255–97.

[33] Gershwin S. Design and operation of manufacturing systems—the control point policy. IIE Trans 2000;32:891–906.

[34] Weiss G. On optimal draining of fluid reentrant lines. In: Kelly F, Williams R, editors. Stochastic networks. IMA volumes in mathematics and its applications, vol. 71. New York: Springer-Verlag; 1995. p. 93–105.

[35] Meyn S. Sequencing and routing in multiclass queueing networks. Part II: workload relaxations. SIAM J Control Optim 2003;42(1):178–217.

[36] Harrison J. Heavy traffic analysis of a system with parallel servers: asymptotic optimality of discrete-review policies. Ann Appl Probab 1998;8(3):822–48.

[37] Hajek B. Optimal control of two interacting service stations. IEEE Trans Automat Control 1984;AC-29:491–9.
[38] Chen RR, Meyn S. Value iteration and optimization of multiclass queueing networks. Queueing Syst Theory Appl 1999;32(1–3):65–97.
[39] Wein L. Scheduling networks of queues: heavy traffic analysis of a multistation network with controllable inputs. Oper Res 1990;40:312–34.
[40] Paschaladis I, Su C, Caramanis M. Target-pursuing scheduling and routing policies for multiclass queueing networks. IEEE Trans Automat Control 2004;49(10):1709–22.
[41] Veatch M. Performance bounds in queueing networks. In: Cochran JJ, Cox LA, Keskinocak JC, Kharoufeh JP, Smith JC, editors. Wiley Encyclopedia of Operations Research and Management Science. Hoboken, NJ: John Wiley and Sons,. Inc.; 2010.
[42] Kumar S, Kumar PR. Performance bounds for queueing networks and scheduling policies. IEEE Trans Automat Control 1994;39:1600–11.
[43] Russell M, Fraser J, Rizzo S, Veatch M. Comparing LP bounds for queueing networks. IEEE Trans Automat Control 2009;54(11):2703–7.
[44] Keller P, Mannor S, Precup D. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In: Proceedings of the 23rd international conference on machine learning. New York: ACM; 2006. p. 449–56.