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.

© 2015 Elsevier Ltd. All rights reserved.

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/© 2015 Elsevier Ltd. All rights reserved.

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 $\rho<\mathbf{1}$.