An Approximate Planning Model for Distributed Computing Networks

John S. Hollywood

RAND Corporation, 1200 South Hayes Street, Arlington, Virginia 22202

Received 28 June 2004; revised 14 April 2005; accepted 8 May 2005
DOI 10.1002/nav.20098
Published online 12 July 2005 in Wiley InterScience (www.interscience.wiley.com).

Abstract: We develop an approximate planning model for a distributed computing network in which a control system oversees the assignment of information flows and tasks to a pool of shared computers, and describe several optimization applications using the model. We assume that the computers are multithreaded, and have differing architectures leading to varying and inconsistent processing rates. The model is based on a discrete-time, continuous flow model developed by Graves [Oper Res 34 (1986), 522–533] which provides the steady-state moments of production and work-in-queue quantities. We make several extensions to Graves' model to represent distributed computing networks. First, we approximately model control rules that are nonlinear functions of the work-in-queue at multiple stations through a linearization approach. Second, we introduce an additional noise term on production and show its use in modeling the discretization of jobs. Third, we model groups of heterogeneous computers as aggregate, "virtual computing cells" that process multiple tasks simultaneously, using a judiciously selected control rule. © 2005 Wiley Periodicals, Inc. Naval Research Logistics 52: 590–605, 2005.

Keywords: computing; computing networks; queuing; queuing networks

1. INTRODUCTION

In this paper, we consider an approximate planning model for *grid-based* distributed computing networks, in which computing resources are shared among a number of separate users and processes. Rather than run jobs on an individual computer, network users submit jobs to a control system that assigns jobs to a pool of available computers. Currently, there are a number of efforts underway to develop grid-based networks, such as the Globus Project (cf. Foster, Kesselman and Tuecke [10]; Frey et al. [11], for resource management; project web site http://www.globus.org) and the Legion Project (cf. Grimshaw, Wulf, et al. [15]; project web site http://www.cs.virginia.edu/~legion/).

The model in this paper analyzes grid-based networks processing sets of *information flows.* An information flow comprises all the processing tasks required to produce an information product (or set of related information products). For example, an "image processing" flow would include all the tasks from initial scanning of the image, through the application of multiple processing algorithms, to displaying the final images on users' monitors. In principle, a grid-based network allows system managers to control the network to maximize the performance of these flows. The model allows flows to have branches and feedback loops, and for the network to process multiple flows simultaneously.

In this paper, work is done by a set of *computing cells.* A cell is a "virtual computer" that processes a set of information flow tasks, with a subset of computing resources dedicated to each task. The implementation of a cell may range from a timeshare of a computer, to a single computer, to a group of computers. Importantly, the computers assigned to the cells may be heterogeneous and even *inconsistent*—implying, for example, that a type 1 computer might process jobs of type A twice as fast as a type 2 computer, but that a type 2 computer might process jobs of type B three times as fast a type 1 computer.

The processing of each type of task by each cell is represented through the use of a virtual *work station;* each cell supports one or more stations processing information flow tasks. Cells are assumed to use a processor-sharing discipline, such that the work performed for each task depends both on the total work performed by the cell and the relative work-in-queue for that task. The model estimates the steady-state expectation and variance of the work done for each station, as well as the steady-state expectation and variance of the work-in-queue for each station. It is possible to aggregate across the station statistics and calcu-

Correspondence to: J.S. Hollywood (johnsh@rand.org)

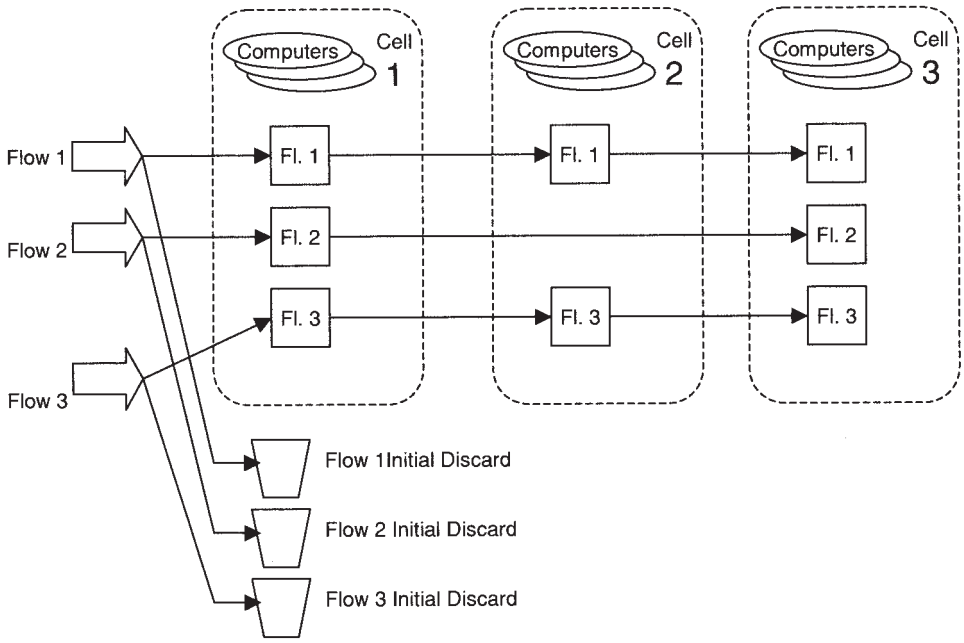

Figure 1. Example model of a distributed computing network with multiple flows.

late estimates for the first two moments of work and work-in-queue for each computing cell. The model can be used for a variety of resource allocation applications; examples are given later in the paper.

Figure 1 shows an example grid-based network, in which three computing cells process the work from three linear information flows in eight stations, simultaneously. The network can accept or discard incoming work for each flow, depending on performance constraints.

The planning model is based on Graves's Tactical Planning Model (TPM [13]). The TPM is a Markovian, discrete-time, continuous flow model to support tactical planning in job shops, in which the work flows can include multiple branches and feedback loops. The planning model expands on the TPM in several major ways.

First, the planning model supports concave control rules in which a computer's ability to respond to increased work diminishes as it becomes more heavily loaded, trending towards an asymptotic maximum. We also permit the control rules to be functions of the work-in-queue at multiple stations; this allows for modeling computing cells that process multiple types of tasks simultaneously. This behavior is similar to that observed on multitasking computers, with the exception that computer performance usually degrades once job loading exceeds critical levels (for examples of this behavior, see Welsh, Culler, and Brewer [26]). We address this exception by assuming that computers incorporate controllers limiting the number of jobs in service; we are familiar with real-world examples where such controllers have been used successfully. Although the resulting model is approximate, we will show that the model is accurate when the network is "reasonably well behaved" (output at each station is at least 20% less than the asymptotic maximum, and the standard deviation of the arriving work is less than its expectation).

In general, little has been done to model job shops with nonlinear control rules, especially stochastically. Karmarkar [20, 21] presents a discrete-time model of a job shop that uses concave control rules called *clearing functions,* but the resulting model is strictly deterministic. A number of researchers have considered an extreme case of a nonlinear control function, the *bounded* control rule, in which each station processes its work-in-queue up to a fixed capacity each time period. The models using bounded control rules are deterministic; examples include the work of Hax [16], Baker [1], and Bitran and Tirupati [4]. Graves [14] considered staffing issues in a repair depot; his analysis related completed work to a piecewise linear function of the work-in-queue. However, the subsequent analysis then approximated the piecewise linear control rule with a strictly linear control rule. One can create conventional queuing networks similar in character to the original TPM (with linear control rules) by having each station be an infinite-server queue; the analysis of such networks was done by Baskett et al. [2] and Kelly [22]. However, the results will differ since infinite-server queues have incoming jobs enter service immediately. One can also, in principle, create conventional queuing networks that model nonlinear control rules by associating greater station service rates with higher numbers of

jobs in queue (cf. Jackson [18, 19]). However, the resulting analysis is difficult.

Second, the model represents groups of computers as aggregate cells that behave as if they were a single computer, and represents cells' processing of multiple computing tasks simultaneously. The computers comprising the cells may be heterogeneous and even inconsistent—important given that different types of computers often show great inconsistencies in the time they require to process jobs. For example, the Standard Performance Evaluation Corporation (SPEC) reports significantly different rankings for its CPU2000 processor speed tests on integer calculations (SPECint2000) vs. floating point calculations (SPECfp2000) [25]. This extension significantly reduces the complexity of network models while capturing the primary behavior of multithreaded computers within the network. The extension depends on a judicious selection of a nonlinear control rule—in this case, Karmarkar's clearing function.

Third, the model supports some relaxation of the TPM's Markovian assumption. The model allows multiple stations per computing cell, with each station potentially being from an entirely different information flow; it is not assumed that the work leaving from each cell is indistinguiable. The Markovian property remains in effect for tasks with each information flow. However, this is usually a reasonable assumption, given that an information flow represents a common process to produce a single type of information product.

Finally, the model supports some relaxation of the assumption of work being a continuous flow. In practice, computing cells frequently process work in discrete jobs rather than as a continuous "fluid"; this increases the variance of the work output at each station. The model represents this effect through the use of a noise term on work output at each station; as an example, the paper shows how to generate approximate lower and upper bounds on these noise terms for most situations.

Section 2 presents the mathematics of the model, extending the TPM. Section 3 applies the resulting model to multithreaded computing cells processing multistage information flows simultaneously. Section 4 models the contributions of individual, heterogeneous computers to computing cells. Section 5 discusses the use of the planning model for optimization, including maximum throughput, minimum work-in-queue, and minimum cost applications. Section 6 presents several examples; the initial examples demonstrate the accuracy of the approximations, and the latter example presents optimization application. Finally, Section 7 discusses the model, its uses, and how it might be extended.

2. THE CORE MODEL

We extend Graves [13] to permit control rules that are nonlinear functions of the queue lengths at multiple stations, plus a noise term on production. We assume that the work produced at station i in time t is given by the control rule:

$$W_{it} = p_i(\mathbf{Q_t}) + \zeta_{it}, \tag{1}$$

where W_{it} is the work produced (the *production*) at station i for period t, Q_{it} is the work-in-queue at the start of period t, $\mathbf{Q_t}$ is the corresponding vector of queue-length quantities for all stations, $p_i(\mathbf{Q_t})$ is a production function that is continuous, monotonic, and twice-differentiable over the ranges of work and work-in-queue quantities seen in the job shop, and ζ_{it} is an independent and identically distributed random variable with finite mean and variance representing production noise. If ζ_{it} has a mean of less than zero, it is assumed that the mean is small enough that W_{it} does not drop below zero. Further, while ζ_{it} is not correlated over time, it may be correlated to the production at other stations in the network.

The production and work-in-queue quantities follow a standard inventory balance equation:

$$Q_{it} = Q_{i,t-1} - W_{i,t-1} + A_{it}, \tag{2}$$

where A_{it} is the work that arrives to station i at the start of period t. There are two types of work arrivals at station i. The first type comprises work arriving at the station from outside the network, and is modeled with an independent and identically random variable. The second type comprises work arriving from upstream stations, and is assumed to be directly proportional to the work completed at upstream stations plus random noise. Thus,

$$A_{it} = \varepsilon_{it} + \sum_j \phi_{ij} W_{j,t-1}. \tag{3}$$

In the above equation, ε_{it} is an independent and identically distributed random variable with a nonnegative mean and a finite variance; it represents work arriving from outside the shop plus any noise in the work arriving from upstream stations. ε_{it} is not correlated over time, but may be correlated with arrivals to other stations in the job shop. ϕ_{ij} is a constant representing the expected work that arrives at station i given one unit of production at station j. The summation represents the total expected work arriving at station i due to upstream production in the previous period. Substituting (3) and (1) into (2) yields the following recursion equation for the queue length at station i:

$$Q_{it} = Q_{i,t-1} - p_i(\mathbf{Q_{t-1}}) - \zeta_{j,t-1} + \varepsilon_{it} + \sum_j \phi_{ij} \times [p_j(\mathbf{Q_{t-1}}) + \zeta_{j,t-1}]. \quad (4)$$

Since $p_i(\mathbf{Q_t})$ is continuous, twice differentiable, and monotonically increasing over the range of possible production and work-in-queue values, $p_i(\mathbf{Q_t})$ has an inverse function, $q_i(\mathbf{W_t})$. By substituting this function into (4), we can rewrite (4) as a recursion equation for the production quantities:

$$q_i(\mathbf{W_t} - \boldsymbol{\zeta_t}) = q_i(\mathbf{W_{t-1}} - \boldsymbol{\zeta_{t-1}}) - W_{i,t-1} + \varepsilon_{it} + \sum_j \phi_{ij}[W_{j,t-1}]. \quad (5)$$

To simplify (5), we let $P_{it} = W_{it} - \zeta_{it}$; P_{it} corresponds to the production as a function of $\mathbf{Q_t}$ prior to adding the noise term. This yields

$$q_i(\mathbf{P_t}) = q_i(\mathbf{P_{t-1}}) - (P_{i,t-1} + \zeta_{it-1}) + \varepsilon_{it} + \sum_j \phi_{ij} \times [P_{i,t-1} + \zeta_{i,t-1}]. \quad (6)$$

We write all of the queue-length recursion equations (4) and all of the production recursion equations (6) for all stations in the job shop simultaneously in matrix-vector form:

Queue Lengths:

$$\mathbf{Q_t} = \mathbf{Q_{t-1}} + (\boldsymbol{\Phi} - \mathbf{I})\mathbf{p}(\mathbf{Q_{t-1}}) + \boldsymbol{\varepsilon_t} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\zeta_{t-1}}, \quad (7)$$

Production Quantities:

$$\mathbf{q}(\mathbf{P_t}) = \mathbf{q}(\mathbf{P_{t-1}}) + (\boldsymbol{\Phi} - \mathbf{I}) \times \mathbf{P_{t-1}} + \boldsymbol{\varepsilon_t} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\zeta_{t-1}},$$

$$\text{where} \qquad \mathbf{P_t} = \mathbf{W_t} - \boldsymbol{\zeta_t}. \quad (8)$$

Here, $\mathbf{P_t}$ is the vector of work production quantities before adding noise terms, $\mathbf{Q_t}$ is the vector of queue lengths, $\boldsymbol{\Phi}$ is the matrix of ϕ_{ij} terms. $\mathbf{p}(\mathbf{Q_t})$ is a vector-valued function whose ith value equals $p_i(\mathbf{Q_t})$, and $\mathbf{q}(\mathbf{P_t})$ is a vector-valued function whose ith value equals $q_i(\mathbf{P_t})$. $\boldsymbol{\varepsilon_t}$ is the vector of the ε_{it}'s, and has mean $\boldsymbol{\mu_\varepsilon}$ and covariance matrix $\boldsymbol{\Sigma_\varepsilon}$. $\boldsymbol{\zeta_{t-1}}$ is the vector of the ζ_{it}'s, and has mean $\boldsymbol{\mu_\zeta}$ and covariance matrix $\boldsymbol{\Sigma_\zeta}$.

Equations (7) and (8) do not lend themselves to direct analysis due to the nonlinearity of the $\mathbf{p}(\mathbf{Q_t})$ and $\mathbf{q}(\mathbf{P_t})$ terms. Instead, we linearize (7) and (8), using the Delta Method (cf. Rice [23], pp. 149–154) to develop first and second order approximations for $\mathrm{E}(\mathbf{q}(\mathbf{P_t}))$ and $\mathrm{var}(\mathbf{q}(\mathbf{P_t}))$. The Delta Method approximates a function of random variables with Taylor series expansions of that function around the means of the random variables. A first-order expansion of (8) around $\bar{\mathbf{P}}_\mathbf{t}$, the expected production before noise is introduced, at time t, is

$$\mathbf{q}(\bar{\mathbf{P}}_\mathbf{t}) + \boldsymbol{\Psi_t} \cdot (\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t}) \approx \mathbf{q}(\bar{\mathbf{P}}_\mathbf{t-1}) + \boldsymbol{\Psi_{t-1}} \cdot (\mathbf{P_{t-1}} - \bar{\mathbf{P}}_\mathbf{t-1}) + (\boldsymbol{\Phi} - \mathbf{I})\mathbf{P_{t-1}} + \boldsymbol{\varepsilon_t} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\zeta_{t-1}}, \quad (9)$$

and the second order expansion is

$$\mathbf{q}(\bar{\mathbf{P}}_\mathbf{t}) + \boldsymbol{\Psi}_t \cdot (\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t}) + \frac{1}{2}\boldsymbol{\Psi}_{2,t}(\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t}) \approx \mathbf{q}(\bar{\mathbf{P}}_\mathbf{t-1}) + \boldsymbol{\Psi}_{t-1} \cdot (\mathbf{P_{t-1}} - \bar{\mathbf{P}}_\mathbf{t-1}) + \frac{1}{2}\boldsymbol{\Psi}_{2,t-1}(\mathbf{P_{t-1}} - \bar{\mathbf{P}}_\mathbf{t-1}) + (\boldsymbol{\Phi} - \mathbf{I})\mathbf{P_{t-1}} + \boldsymbol{\varepsilon_t} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\zeta_{t-1}}, \quad (10)$$

where

- $\bar{\mathbf{P}}_\mathbf{t}$ is the vector of expected production at time t before adding the noise term.
- $\boldsymbol{\Psi}_t$ is a matrix whose ijth element is $\partial q_i(\bar{\mathbf{P}}_\mathbf{t})/\partial P_{jt}$.
- $\boldsymbol{\Psi}_{2,t}(\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t})$ is a vector whose ith element is $(\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t})^\mathbf{T}\boldsymbol{\Psi_{2i}}(\mathbf{P_t} - \bar{\mathbf{P}}_\mathbf{t})$, where the jkth element of matrix $\boldsymbol{\Psi_{2i}}$ is $\partial^2 q_i(\bar{\mathbf{P}}_\mathbf{t})/\partial P_{jt}\partial P_{kt}$.

Then the expectation of the second-order approximation is

$$\mathbf{q}(\bar{\mathbf{P}}_\mathbf{t}) + \frac{1}{2}\boldsymbol{\Psi}_{2,t}(\mathrm{var}(\mathbf{P_t})) \approx \mathbf{q}(\bar{\mathbf{P}}_\mathbf{t-1}) + \frac{1}{2}\boldsymbol{\Psi}_{2,t-1}(\mathrm{var}(\mathbf{P_{t-1}})) + (\boldsymbol{\Phi} - \mathbf{I})\bar{\mathbf{P}}_\mathbf{t-1} + \boldsymbol{\mu_\varepsilon} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\mu_\zeta}, \quad (11)$$

where $\boldsymbol{\Psi}_{2,t}(\mathrm{var}(\mathbf{P_t}))$ is a vector whose ith element is

$$\sum_j \sum_k \frac{\partial^2 q_i(\bar{P}_t)}{\partial P_{jt}\partial P_{kt}} \mathrm{cov}(P_{jt}, P_{kt}),$$

such that $\mathrm{cov}(P_{jt}, P_{kt})$ is the jkth element of $\mathrm{var}(\mathbf{P_t})$. Since $\mathrm{var}(\mathbf{P_t})$ will not be known exactly, we will use the first-order approximation described below.

The variance of the first-order approximation is

$$\boldsymbol{\Psi_t} \cdot \mathrm{var}(\mathbf{P_t}) \cdot \boldsymbol{\Psi}_\mathbf{t}^\mathbf{T} \approx (\boldsymbol{\Phi} - \mathbf{I} + \boldsymbol{\Psi_{t-1}}) \cdot \mathrm{var}(\mathbf{P_{t-1}}) \cdot (\boldsymbol{\Phi} - \mathbf{I} + \boldsymbol{\Psi_{t-1}})^T + \boldsymbol{\Sigma_\varepsilon} + (\boldsymbol{\Phi} - \mathbf{I}) \cdot \boldsymbol{\Sigma_\zeta} \cdot (\boldsymbol{\Phi} - \mathbf{I})^T. \quad (12)$$

Equations (11) and (12) thus give approximate recursion equations for the production moments. It is also possible to

find approximate recursion equations for the queue length moments by applying the Delta Method to (7).

To approximate the steady-state moments, we find stationary solutions to Eqs. (11) and (12). We begin with expected production before noise terms are added. Assuming that a steady-state vector for $\bar{\mathbf{P}}$ exists, Eq. (11) reduces in steady-state to

$$0 = (\mathbf{\Phi} - \mathbf{I})\bar{\mathbf{P}} + \boldsymbol{\mu}_{\boldsymbol{\varepsilon}} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\mu}_{\boldsymbol{\zeta}}, \tag{13}$$

and solving for $\bar{\mathbf{P}}$ immediately yields

$$\bar{\mathbf{P}} = (I - \mathbf{\Phi})^{-1}[\boldsymbol{\mu}_{\boldsymbol{\varepsilon}} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\mu}_{\boldsymbol{\zeta}}]. \tag{14}$$

Equation (14) is this model's equivalent of standard balance equations. It holds regardless of the control rule used, assuming the control rule used results in the shop being stable.

We now turn to the steady-state production variance. In steady state, the expected queue lengths and production quantities are the same in all periods, so the first-order approximation of the production recursion Eq. (9) becomes

$$\mathbf{q}(\bar{\mathbf{P}}) + \mathbf{\Psi} \cdot (\mathbf{P_t} - \bar{\mathbf{P}}) \approx \mathbf{q}(\bar{\mathbf{P}}) + \mathbf{\Psi} \cdot (\mathbf{P_{t-1}} - \bar{\mathbf{P}}) + (\mathbf{\Phi} - \mathbf{I}) \times \mathbf{P_{t-1}} + \boldsymbol{\varepsilon}_\mathbf{t} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\zeta}_{\mathbf{t-1}}, \tag{15}$$

where $\mathbf{\Psi}$ is a matrix whose ijth element is $\partial q_i(\bar{\mathbf{P}})/\partial P_j$. Simplifying (15) and solving the resulting equation for $\mathbf{P_t}$ yields

$$\mathbf{P_t} = (\mathbf{I} - \mathbf{\Psi^{-1}} + \mathbf{\Psi^{-1}\Phi})\mathbf{P_{t-1}} + \mathbf{\Psi^{-1}}[\boldsymbol{\varepsilon}_\mathbf{t} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\zeta}_{\mathbf{t-1}}]. \tag{16}$$

Taking the variance of (16) yields a recursion equation for $\mathrm{var}(\mathbf{P_t})$:

$$\mathrm{var}(\mathbf{P_t}) = \mathbf{B} \cdot \mathrm{var}(\mathbf{P_{t-1}}) \cdot \mathbf{B^T} + \mathbf{\Psi^{-1}}\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}(\mathbf{\Psi^{-1}})^\mathbf{T} + (\mathbf{B} - \mathbf{I}) \times \boldsymbol{\Sigma}_{\boldsymbol{\zeta}}(\mathbf{B} - \mathbf{I})^\mathbf{T},$$

$$\text{where} \qquad \mathbf{B} = \mathbf{I} - \mathbf{\Psi^{-1}} + \mathbf{\Psi^{-1}\Phi}. \tag{17}$$

Iterating this equation infinitely yields the desired first-order estimate for $\mathrm{var}(\mathbf{P})$:

$$\mathrm{var}(\mathbf{P}) \approx \sum_{s=0}^{\infty} \mathbf{B}^s(\mathbf{\Psi^{-1}}\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}(\mathbf{\Psi^{-1}})^\mathbf{T} + (\mathbf{B} - \mathbf{I})\boldsymbol{\Sigma}_{\boldsymbol{\zeta}}(\mathbf{B} - \mathbf{I})^\mathbf{T})\mathbf{B}'^s,$$

$$\text{where} \qquad \mathbf{B} = \mathbf{I} - \mathbf{\Psi^{-1}} + \mathbf{\Psi^{-1}\Phi}. \tag{18}$$

The power series may be evaluated numerically or analytically using the techniques discussed in Graves [13]. In the examples that follow, we use Graves's numerical technique. Let $\mathbf{S_n}$ be the sum of the first n terms of (18). Then we have the following recursion:

$$\mathbf{S_{2n}} = \mathbf{S_n} + \mathbf{B^n} \cdot \mathbf{S_n} \cdot (\mathbf{B^T})^\mathbf{n}. \tag{19}$$

Iterating this expression rapidly leads to an accurate value for $\mathbf{S_n} = \mathrm{var}(\mathbf{P})$.

For (19) to converge, $\mathbf{B}$ must have a spectral radius less than one. Graves [13] showed that an expression identical in form to (19) will converge provided that the spectral radius of $\mathbf{\Phi}$ is less than 1, and that $\mathbf{\Psi}$ is a positive matrix. The latter condition will be met for most inverse production functions, given that it just requires $\partial q_i(\bar{\mathbf{P}})/\partial P_j \geq 0$, and $\partial q_i(\bar{\mathbf{P}})/\partial P_i > 0$, for all stations i and j.

To calculate the moments of $\mathbf{W}$, the actual work produced at each station, from the moments of $\mathbf{P}$, we simply add the steady-state noise term for production to $\mathbf{P}$ and calculate the moments:

$$\bar{\mathbf{W}} = \bar{\mathbf{P}} + \boldsymbol{\mu}_{\boldsymbol{\zeta}},$$

$$\mathrm{var}(\mathbf{W}) = \mathrm{var}(\mathbf{P}) + \boldsymbol{\Sigma}_{\boldsymbol{\zeta}}, \tag{20}$$

Here, the calculation of the expectation and variance is direct since ζ_t, as the exogenous noise term for completed work, is independent from $\mathbf{P_t}$ in any period.

We now use the production moments to estimate the queue length moments. Applying the Delta Method to the expression $\mathbf{Q_t} = \mathbf{q}(\mathbf{P_t})$, the second order-estimate of each $E(Q_i)$, given estimates of $\mathrm{E}(\mathbf{P})$ and $\mathrm{var}(\mathbf{P})$, is

$$E(Q_i) \approx q_i(\bar{P}_i) + \frac{1}{2}\sum_j \sum_k \frac{\partial^2 q_i(\bar{\mathbf{P}})}{\partial P_{ji}\partial P_{kt}} \mathrm{cov}(P_j, P_k). \tag{21}$$

We use the second-order estimate of $\mathrm{E}(\mathbf{Q})$ to estimate $\mathrm{var}(\mathbf{Q})$. Applying the Delta Method to (7), a first-order approximation of the queue-length recursion equation is

$$\mathbf{Q_t} \approx \mathbf{Q_{t-1}} + (\mathbf{\Phi} - \mathbf{I})[\mathbf{p}(\bar{\mathbf{Q}}) + \mathbf{\Pi} \cdot (\mathbf{Q_{t-1}} - \bar{\mathbf{Q}})] + \boldsymbol{\varepsilon}_\mathbf{t} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\zeta}_\mathbf{t}, \tag{22}$$

where $\mathbf{\Pi}$ is a matrix whose ijth entry is given by $\partial p_i(\bar{\mathbf{Q}})/\partial Q_j$. The variance of the above approximation is

$$\mathrm{var}(\mathbf{Q_t}) = \mathbf{B} \cdot \mathrm{var}(\mathbf{Q_{t-1}}) \cdot \mathbf{B^T} + \boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} + (\mathbf{\Phi} - \mathbf{I})\boldsymbol{\Sigma}_{\boldsymbol{\zeta}}(\mathbf{\Phi} - \mathbf{I})^\mathbf{T},$$

$$\text{where} \qquad \mathbf{B} = \mathbf{I} + (\mathbf{\Phi} - \mathbf{I}) \cdot \mathbf{\Pi}. \tag{23}$$

Iterating the above expression infinitely yields the desired first-order estimate for var(**Q**):

$$\text{var}(\mathbf{Q}) \approx \sum_{s=0}^{\infty} \mathbf{B}^s(\mathbf{\Sigma}_{\boldsymbol{\varepsilon}} + (\mathbf{\Phi} - \mathbf{I})\mathbf{\Sigma}_{\boldsymbol{\xi}}(\mathbf{\Phi} - \mathbf{I})^{\mathbf{T}})(\mathbf{B}^{\mathbf{T}})^s,$$

$$\text{where} \qquad \mathbf{B} = \mathbf{I} + (\mathbf{\Phi} - \mathbf{I}) \cdot \mathbf{\Pi}. \tag{24}$$

2.1. Discussion

(1) As with other Taylor-series expansions, the accuracy of the above approximations depends on the form of the control rule. If the control rule can be reasonably approximated around the mean with quadratic functions (for the expected queue lengths) and linear functions (for the production and queue length variances) the approximations should be accurate. Hollywood [17] performed simulation analysis of single and multiple-station networks using a class of nonlinear control rules called *clearing functions* (see Section 3 below), which are concave, monotonically increasing functions of work-in-queue tending towards an asymptotic limit; the detailed results (for single-station networks) are given in the first example in Section 6. In brief, expected queue lengths were within 4%, and variance estimates were within 15%, of the simulation results, provided that the expected production was less than 5/6 of the asymptotic maximum output of a station, and the coefficient of variation of the input streams was less than one. The first example in Section 6 explores the accuracy of the approximations for a single station model.

(2) The calculations of the completed work and work-in-queue moments for a computing cell follow directly from the above approximations. The cell's expectations are simply the sum of the expectations of the cell's stations. The cell's variances are the sum of the variances of the cell's stations, plus the sum of all the covariance terms between the cell's stations.

(3) The appropriate size of a period depends on the nature of the network being modeled. The general rule, dating back to Graves [13], is that the periods should be large enough that there is likely to be some work arriving in each period, but small enough that jobs are unlikely to move through more than one station during a period. Thus, for example, if the computer network tends to have a few large jobs arriving hourly, half-hour or hour periods will be appropriate; conversely, if significant numbers of small jobs arrive every second, seconds are appropriate periods.

(4) The approximations assume that the network has a steady state, which is often not the case. To address networks that do not have a steady state, one approach is to note major phases of operation for the network during which work arrivals substantially differ (e.g., network traffic during peak hours as opposed to traffic at night, or network traffic during surges) and run the model separately for each phase.

2.2. Modeling of Discrete Jobs

As noted, the use of a noise term for production allows for the modeling of stations completing discrete jobs instead of a work "fluid." When this extension is used, each period can be treated as a snapshot in time, tracking the total number of jobs that are completed and move downstream in each period.

This increases the variance of production somewhat, as the actual work completed each period depends on whether the corresponding jobs finish in a given period, as opposed to being a smooth function of the total work in queue. The total work in queue at station i, Q_{it}, may be thought of as a random number of independent, identically distributed random variables, where the sum is n_{it}, the total number of jobs in the queue, and the variables represent the length of each job in the queue, N_{ikt}. Assume that the length of each job at station i is independent and identically distributed with expectation $E(N_i)$ and variance $\text{var}(N_i)$. We have that

$$Q_{it} = q(P_{it}) = \sum_{k=1}^{n_{it}} N_{ikt}. \tag{25}$$

We can use the standard formulas for the moments of a random sum of independent, identically distributed random variables (cf. Drake [8], pp. 111–112; formulas follow from the use of transforms), to find that

$$E(Q_{it}) = E(n_{it})E(N_i),$$

$$\text{var}(Q_{it}) = E(n_{it})\text{var}(N_i) + [E(N_i)]^2\text{var}(n_{it}). \tag{26}$$

The actual work produced in period t, W_{it}, is a subset of the job lengths in (25), where the subset derives from those jobs completed in period t.

We consider two methods for approximating the moments of W_{it} for cases in which P_{it} is a function of single queue-length Q_{it} only. The methods derive from two different assumptions about which jobs are completed, and are likely to be lower and upper bounds in practice.

Case 1 (*Typical lower bound*): We assume that a given proportion of the jobs in queue are completed in each time period, which is a reasonable assumption if the jobs are reasonably small and the variance in job length is low. The natural choice is to assume that $P_{it}/Q_{it} = P_{it}/q(P_{it})$ of the jobs will be completed. To make the subsequent equations

tractable, we substitute the first-order approximation $Q_{it} \approx L_i P_{it}$, where $q'(\bar{P}_i) = L_i$; this approximation fixes the proportion of jobs in queue completed in each time period to be $1/L_i$. We have

$$W_{it} \approx \sum_{k=1}^{n_{it}/L_i} N_{ikt}, \tag{27}$$

which, applying the formulas in (26), yields the following approximate moments:

$$E(W_{it}) \approx L_i^{-1} E(n_{it}) E(N_i),$$

$$\operatorname{var}(W_{it}) \approx L_i^{-1} E(n_{it}) \operatorname{var}(N_i) + L_i^{-2} [E(N_i)]^2 \operatorname{var}(n_{it}). \tag{28}$$

We use (26) to write (28) in terms of Q_{it} rather than n_{it}:

$$E(W_{it}) \approx L_i^{-1} E(Q_{it}),$$

$$\operatorname{var}(W_{it}) \approx L_i^{-1} E(Q_{it}) E(N_i)^{-1} \operatorname{var}(N_i) + (L_i^{-2} \operatorname{var}(Q_{it}) - L_i^{-2} E(N_i)^{-1} E(Q_{it}) \operatorname{var}(N_i)). \tag{29}$$

Continuing the use of the first-order approximation $Q_{it} \approx L_i P_{it}$, and simplifying terms, we have

$$E(W_{it}) \approx E(P_{it}),$$

$$\operatorname{var}(W_{it}) \approx \operatorname{var}(P_{it}) + [(1 - L_i^{-1}) E(P_{it}) E(N_i)^{-1} \operatorname{var}(N_i)]. \tag{30}$$

The approximation sets $E(W_{it}) = E(P_{it})$ as expected, given that discretizing the work flow does not change the total work performed. Importantly, the approximation separates $\operatorname{var}(W_{it})$ into a term that depends strictly on $\operatorname{var}(P_{it})$, and a term that depends strictly on $\operatorname{var}(N_{it})$, with the latter term being a constant in steady state. The additional variance term tends to zero as the variance of job lengths decreases, which is as expected.

By matching moments with (30), we model W_{it} as

$$W_{it} = P_{it} + \zeta_{it},$$

$$E(\zeta_{it}) = 0,$$

$$\operatorname{var}(\zeta_{it}) = [(1 - L_i^{-1}) E(P_i) E(N_i)^{-1} \operatorname{var}(N_i)]. \tag{31}$$

Case 2 (*Typical upper bound*): We assume that the endpoints of the jobs are uniformly distributed throughout the queue, in which case each job has a P_{it}/Q_{it} probability of being completed during the course of the period. This is a reasonable assumption when the jobs are comparatively large and job lengths have a significant variance. In fact, the job endpoints will be uniformly distributed throughout the queue when the job lengths are exponentially distributed (cf. Gallager [12], pp. 45–48), in which case the jobs have variance $E(N_i)^2$—likely to be the noisiest job types encountered in practice, making this approximation a typical upper bound.

Again using the first order assumption $Q_{it} \approx L_i P_{it}$, we have that each job has approximately a $1/L_i$ probability of being completed. We have

$$W_{it} \approx \sum_{k=1}^{n_{it}} \sum_{k=1}^{I_{ikt}} N_{ikt}, \tag{32}$$

where I_{ikt} is a Bernoulli random variable with parameter $1/L_i$. The expectation of I_{ikt} is $1/L_i$, and the variance is $(1/L_i)(1 - 1/L_i)$. We iterate (26) to find the moments of (32):

$$E(W_{it}) \approx E(n_{it}) L_i^{-1} E(N_i),$$

$$\operatorname{var}(W_{it}) \approx E(n_{it}) L_i^{-1} \operatorname{var}(N_i) + E(n_{it}) [E(N_i)]^2 (L_i^{-1} - L_i^{-2}) + [E(N_i)]^2 L_i^{-2} \operatorname{var}(n_{it}). \tag{33}$$

Writing (33) in terms of Q_{it}, substituting the approximation $Q_{it} \approx L_i P_{it}$, and simplifying yields

$$E(W_{it}) \approx E(P_{it}),$$

$$\operatorname{var}(W_{it}) \approx \operatorname{var}(P_{it}) + [(1 - L_i^{-1}) E(P_{it}) E(N_i)^{-1} \operatorname{var}(N_i)] + (1 - L_i^{-1}) E(P_{it}) E(N_i). \tag{34}$$

This is the same as the lower bound approximation, (30), with the exception of the third term on $\operatorname{var}(W_{it})$. The third term reflects the impact of the probabilistic completion of jobs; it is usually significantly larger than the second term except when $\operatorname{var}(N_i)$ "approaches exponential" [e.g., $\operatorname{var}(N_i) \approx E(N_i)^2$]. This third term tends to hamper the use of longer queue times to smooth workflow, as significant variance tends to remain simply from the probabilistic processing of the jobs. By matching moments with (34), we model W_{it} as

$$W_{it} = P_{it} + \zeta_{it},$$

$$E(\zeta_{it}) = 0,$$

$$\operatorname{var}(\zeta_{it}) = [(1 - L_i^{-1}) E(P_i) E(N_i)^{-1} \operatorname{var}(N_i)] + (1 - L_i^{-1}) \times E(P_i) E(N_i). \tag{35}$$

Hollywood [17] discusses the modeling of probabilistic job completion rules in some detail, to include analysis of a more general class of control rules in which completed jobs stochastically generate new jobs downstream.

3. MODELING COMPUTING CELLS AND INFORMATION FLOWS

In this section, we apply the core model to processing networks such as that shown in Figure 1. Recall that the processing network is divided into computing cells, with each cell pooling the resources of multiple computers or computer timeshares. Each cell processes multiple types of tasks simultaneously, and the work a cell performs on each type of task is modeled as an individual workstation. The total work the cell performs each period is a nonlinear function of the total work-in-queue from all tasks. The work performed for each task type is proportional to the task-type's share of the cell's total work-in-queue (i.e., the cell does not independently process tasks of each type).

For the sake of simplicity, for the remainder of this paper we assume that work is well approximated as a fluid flow, so that modeling the discretization of jobs is unnecessary. The subsequent discussion continues to use a noise term, $\boldsymbol{\zeta}_\mathbf{t}$, on work production, and so permits the use of discrete-job modeling if necessary.

The control use in this paper is a generalization of the *clearing function,* considered by Karmarkar [20, 21]. With clearing functions, production is a concave, monotonically increasing, function of work-in-queue that approaches an asymptotic maximum. Let the stations representing a cell's work on each type of task be numbered from 1 to K. The total prenoise work a cell performs each period, $P_{cell,t}$, is

$$P_{cell,t} = \frac{M_{cell} \cdot \sum_{k=1}^{K} \alpha_k Q_{kt}}{\sum_{k=1}^{K} \alpha_k Q_{kt} + \beta_{cell}}, \tag{36}$$

where Q_{kt} is station k's queue length at the start of period t, α_k is a smoothing parameter that, roughly speaking, determines the fraction of task type-k work that the cell addresses each period, M_{cell} is an upper bound on the cell's total throughput per period; throughput approaches M_{cell} as the work-in-queue approaches infinity. β_{cell} is a parameter governing the rate at which the work completed per period approaches M_{cell}. We will consider a simple approach for modeling multiple heterogeneous computers each contributing to M_{cell} and β_{cell} a bit later in this paper.

Given an actual network with a history of actual work-in-queue and performance statistics, the above parameters may be interpolated using regression analysis. As an example, Fine and Graves [9] fitted Graves' original Tactical Planning Model to production at a mainframe subcomponents manufacturing plant.

The prenoise work the cell completes for tasks of type j in time t, P_{jt}, equals the percentage of $P_{cell,t}$ corresponding to the amount of type-j work in queue addressed by the cell:

$$P_{jt} = p_j(\mathbf{Q_t}) = \frac{\alpha_j Q_{jt}}{\sum_{k=1}^{K} \alpha_k Q_{kt}} \cdot \frac{M \cdot \sum_{k=1}^{K} \alpha_k Q_{kt}}{\sum_{k=1}^{K} \alpha_k Q_{kt} + \beta} = \frac{M \cdot \alpha_j Q_{jt}}{\sum_{k=1}^{K} \alpha_k Q_{kt} + \beta}. \tag{37}$$

For the sake of simplicity, we have dropped the "cell" subscripts from M and β in (37). We apply the core model's formulas to the family of nonlinear, multistation control rules defined by (37).

Expected Production. The calculation of prenoise work production is direct; from (14), $\bar{\mathbf{P}}_\mathbf{t} = (I - \boldsymbol{\Phi})^{-1}[\boldsymbol{\mu}_\boldsymbol{\varepsilon} + (\boldsymbol{\Phi} - \mathbf{I})\boldsymbol{\mu}_\boldsymbol{\zeta}]$.

Variance of Production. The calculation of var($\mathbf{P}$) requires finding the first derivatives of the inverse production functions. Solving for Q_{jt} in (37), the inverse production function for station j is

$$q_j(\mathbf{P}_t) = \frac{\beta P_{jt}}{\alpha_j (M - \sum_{k=1}^{K} P_{kt})}. \tag{38}$$

In solving (38), we first solve (37) to find a substitute for $\sum_{k=1}^{K} \alpha_k Q_{kt}$ in terms of $P_{cell,t}$, and then use the fact that the cell's total production, $P_{cell,t}$, equals $\sum_{k=1}^{K} P_{kt}$.

Evaluating (38) at E($\mathbf{P}$) yields a first-order approximation for E($\mathbf{Q}$):

$$q_j(\bar{\mathbf{P}}) = \frac{\beta \bar{P}_j}{\alpha_j (M - \sum_{k=1}^{K} \bar{P}_k)}. \tag{39}$$

We compute the first partial derivatives of (39), evaluated at E($\mathbf{P}$):

$$\frac{\partial q_j(\bar{\mathbf{P}})}{\partial P_{jt}} = \frac{\beta (M + \bar{P}_j - \sum_{k=1}^{K} \bar{P}_k)}{\alpha_j (M - \sum_{k=1}^{K} \bar{P}_k)^2}, \tag{40}$$

$$\frac{\partial q_j(\bar{\mathbf{P}})}{\partial P_{kt}} = \frac{\beta \bar{P}_j}{\alpha_j (M - \sum_{k=1}^{J} \bar{P}_k)^2}, \qquad \forall\, k \neq j. \tag{41}$$

We then apply (17), the power series equation for var($\mathbf{P}$) directly, such that the jkth element of the $\boldsymbol{\Psi}$ matrix is $\boldsymbol{\Psi}_{jk} = \partial q_j(\bar{\mathbf{P}})/\partial P_k$.

Expected Queue Lengths. From (21), a second-order estimate for $E(Q_j)$ is

$$E(Q_j) = q_j(\bar{\mathbf{P}}) + \frac{1}{2}\sum_k \sum_l \frac{\partial^2 q_j(\bar{\mathbf{P}})}{\partial P_k \partial P_l} \cdot \mathrm{cov}(P_k, P_l), \quad (42)$$

where the covariance arguments are the (k, l) elements of the first-order estimate of var($\mathbf{P}$), found above, and

$$\frac{\partial^2 q_j(\bar{P})}{\partial P_j^2} = \frac{2\beta(M + \bar{P}_j - \sum_{k=1}^K \bar{P}_k)}{\alpha_j(M - \sum_{k=1}^K \bar{P}_k)^3}, \qquad \text{for } k = j \text{ and } l = j, \quad (43)$$

$$\frac{\partial^2 q_j(\bar{P})}{\partial P_k \partial P_l} = \frac{\beta(M + 2\bar{P}_j - \sum_{k=1}^K \bar{P}_k)}{\alpha_j(M - \sum_{k=1}^K \bar{P}_k)^3}, \qquad \text{for } k \text{ or } l = j, \quad (44)$$

$$\frac{\partial^2 q_j(\bar{P})}{\partial P_k \partial P_l} = \frac{2\beta\bar{P}_j}{\alpha_j(M - \sum_{k=1}^K \bar{P}_k)^3}, \quad \text{for } k \text{ and } l \text{ both not equal to } j. \quad (45)$$

Queue Length Variances. The final step is to calculate a first-order estimate for var($\mathbf{Q}$), which involves computing the first derivatives of the production functions evaluated at E($\mathbf{Q}$). We have

$$\frac{\partial p_j(\bar{\mathbf{Q}})}{\partial P_{jt}} = \frac{M\alpha_j(\sum_{k=1}^K \alpha_k\bar{Q}_k - \alpha_j\bar{Q}_j + \beta)}{(\sum_{k=1}^K \alpha_k\bar{Q}_k + \beta)^2}, \quad (46)$$

$$\frac{\partial p_j(\bar{\mathbf{Q}})}{\partial P_{kt}} = \frac{-M\alpha_k\alpha_j\bar{Q}_j}{(\sum_{k=1}^K \alpha_k\bar{Q}_k + \beta)^2}, \qquad \forall\, k \neq j. \quad (47)$$

We then apply (24), the power series for var($\mathbf{Q}$) directly, such that the jkth element of the $\mathbf{\Pi}$ matrix is $\mathbf{\Pi}_{jk} = \partial q_j(\bar{\mathbf{Q}})/\partial Q_{kt}$.

4. MODELING THE IMPACTS OF INDIVIDUAL COMPUTERS ON INFORMATION FLOWS

In this section, we approximately model the impact of individual computers on information flows. As noted in the introduction, we permit computers to have inconsistent processing rates on different types of jobs. To model this heterogeneity, let m_{ij} be the marginal increase to a cell's maximum service rate, M_{cell}, that results from devoting one type-i computer to type-j tasks. Let y_{ij} be the number of type-i computers assigned to type-j jobs; we permit y_{ij} to be fractional, corresponding to timeshares of type-i computers. Then, given an assignment of computers to tasks within the cell, we use the linear approximation that M_{cell} is the sum of the marginal contributions of each assigned computer, such that $M_{cell} \approx \sum_{i,j} m_{ij}y_{ij}$. The approximation assumes that the computers in the cell collectivity behave as one large, virtual computer. The use of marginal (linear) contributions to M_{cell} is reasonable, since the resulting production formulas show the desired nonlinearity as a function of M_{cell}.

There is supporting evidence to justify this approximation. Under certain conditions, the cell of computers behaves exactly like a single computer. First, assume the job types are consistent, such that $m_{ij} = m_i$ for all job types. Then the work-in-queue on computer k can be represented as a single queue length, Q_{kt}, since the computer responds to work from each type of job in the same way. Second, assume the use of a load-balancing scheduling policy such that the amount of work-in-queue on each computer is proportional to the computer's asymptotic service rate, M_k (where M_k is some m_i), i.e., $Q_{kt} = (M_k/\sum_{l\in cell} M_l)Q_{cell,t}$, where $Q_{cell,t}$ is the total amount of work-in-queue in the cell. Finally, assume that for each computer, parameter $\beta_k = \alpha M_k$, where α is identical for all computers. Then the total amount of prenoise work produced by the cell is

$$P_{cell,t} = \sum_k P_{kt} = \sum_k \frac{M_k Q_{kt}}{Q_{kt} + \beta_i}, \quad (48)$$

which, after substituting for Q_{kt} and β_k as described above and simplifying the result, is

$$P_{cell,t} = \frac{(\sum_k M_k)Q_t}{Q_t + \alpha \cdot \sum_k M_k}, \quad (49)$$

which is equivalent to the production function for a single computer with maximum service rate $M = \sum_k M_k$ and parameter $\beta = \alpha \sum_k M_k$.

5. USING THE MODEL FOR APPROXIMATE OPTIMIZATION

In this section, we employ the model's moment-approximation equations in nonlinear programs to find computing resource allocations that have maximum performance given a budget, or that have minimum cost given performance requirements. For example, a minimum cost program might employ the following variables:

- M_j is the maximum production rate at station j. For the sake of simplicity, we assume that stations represent distinct computing tasks, as opposed to being cells processing multiple computing tasks. The vector of maximum production quantities is $\mathbf{M}$.

- y_{ij} is number of computers of type i assigned to station j. We allow y_{ij} to be fractional, since the network's computers are assumed to be multithreaded. $\mathbf{y}$ is the vector of computer assignments.
- $\mathbf{c}$ is a cost vector for the computers; the objective of a minimum cost problem is to minimize $\mathbf{c} \cdot \mathbf{y}$.
- m_{ij} is the marginal amount by which adding one type-i computer to station j increases the asymptotic maximum production rate (M_j) at station j. The maximum production rate at each station must be less than the total contribution of the station's computers, or $M_j \leq \sum_i m_{ij} y_{ij}$, for all stations j.
- $\mathbf{w_k}$ is a vector of weights on the expected queue lengths. A performance constraint requires a weighted sum of expected queue lengths to be less than some constant, W_k.
- $\boldsymbol{\mu_\varepsilon}$, $\boldsymbol{\mu_\zeta}$, $\boldsymbol{\Sigma_\varepsilon}$, $\boldsymbol{\Sigma_\zeta}$ represent the mean and covariance matrix of the network's exogenous work arrivals and production noise terms, respectively. The resulting nonlinear program is:

$$\begin{aligned} & \min \quad \mathbf{c} \cdot \mathbf{y} \\ \text{subject to} \quad & \mathbf{w_k} \cdot \mathbf{E}\left\lfloor \mathbf{Q}(\mathbf{M}, \boldsymbol{\mu_\varepsilon}, \boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\varepsilon}, \boldsymbol{\Sigma_\zeta}) \right\rfloor \\ & \qquad\qquad \leq W_k, \qquad \text{all } k \\ & M_j \leq \sum_i m_{ij} y_{ij}, \qquad \text{all } j \\ & M_j, y_{ij} \geq 0, \qquad \text{all } i, j, \end{aligned} \tag{50}$$

where $\mathbf{E}\left\lfloor\ \mathbf{Q}(\mathbf{x}, \boldsymbol{\mu_\varepsilon}, \boldsymbol{\mu_\zeta}, \boldsymbol{\Sigma_\varepsilon}, \boldsymbol{\Sigma_\zeta})\ \right\rfloor$ is the vector of approximate expected queue lengths resulting from applying the model with maximum service rate vector $\mathbf{M}$. The above program can be adapted directly to weighted expected waiting times through the use of Little's Law. Maximum performance programs are similar to (50), with the model's expected waiting times appearing in the objective function rather than as a constraint.

It is straightforward to write nonlinear programs in which the computers contribute to the maximum service rate of a cell producing multiple tasks; in such programs, the computer assignments and the cells' processing shares devoted to each task (the α's discussed in Section 3) are both decision variables.

Hollywood [17] describes the solving of nonlinear programs similar to (50) in detail, using the Method of Moments algorithm, which is a standard technique for solving nonlinear programs (described in Bertsekas [3]). This algorithm minimizes a series of augmented Lagrangian functions derived from the nonlinear program (the functions increasingly penalize constraint violations rather than require they be satisfied). It can be shown that the Method of Moments converges to a local minimum, and that the solution found is optimal if the constraints are convex and objective function are both convex (Bertsekas [3]). If the nonlinear programs involve concave control rules, such as the clearing functions emphasized in this paper, the inverse production functions will be convex. Thus, the expressions for the expected queue lengths will also be convex functions, meaning that the solutions to these nonlinear programs will be globally optimal.

To minimize each of the augmented Lagrangian functions, we use the technique of Gradient Projection with Diagonal Scaling, with the Armijo Rule used to determine stepsizes (all techniques, cf. Bertsekas [3]). The biggest bottleneck is repeatedly calculating the gradients and principal second derivatives of the augmented Lagrangian functions (the latter are used in diagonal scaling). Either the objective function or the constraints will be functions of the expected queue lengths, which in turn are functions of the production covariance matrix. The latter is calculated by a matrix power series, making it difficult to find the partial derivatives of the augmented Lagrangians directly. Thus, the gradients and principal second derivatives are calculated numerically, using difference formulas. The difference formulas require two evaluations of the augmented Lagrangian (which requires computing two covariance matrices) for each first and second partial derivative. Consequently, every iteration of the nonlinear optimization algorithm requires computing $O(n)$ covariance matrices. This is a significant bottleneck, hindering an ability to optimize large networks.

Consequently, if one needs to solve a nonlinear program for a particular large model frequently, one may be better off trying to calculate the derivatives of the augmented Lagrangian exactly (using the program MAPLE, for instance). For smaller models that will not be solved often, however, calculating the derivatives likely will not be worth the effort. For example, using custom code in MATLAB, Hollywood [17] reported being able to optimize moderately sized networks (10–15 stations) in well under a minute of computing time on a SPARCstation Classic.

6. EXAMPLES

We first examine the accuracy of the model approximations, first applying the planning model to a single-station, and second to a mainframe subcomponents manufacturing plant. We next consider optimization applications, first minimizing the cost of performance in the same manufacturing plant, and then creating a tradeoff curve between performance and available resources.

Example 1: Estimation Accuracy of a Single-Station Model. We consider 100 test scenarios of a single station (supported by a single cell) using a clearing function and receiving work from a Gamma distribution with an expectation of 10 units. Each test scenarios employs one of 10 standard deviation settings ranging from 0.5 to 15 and one

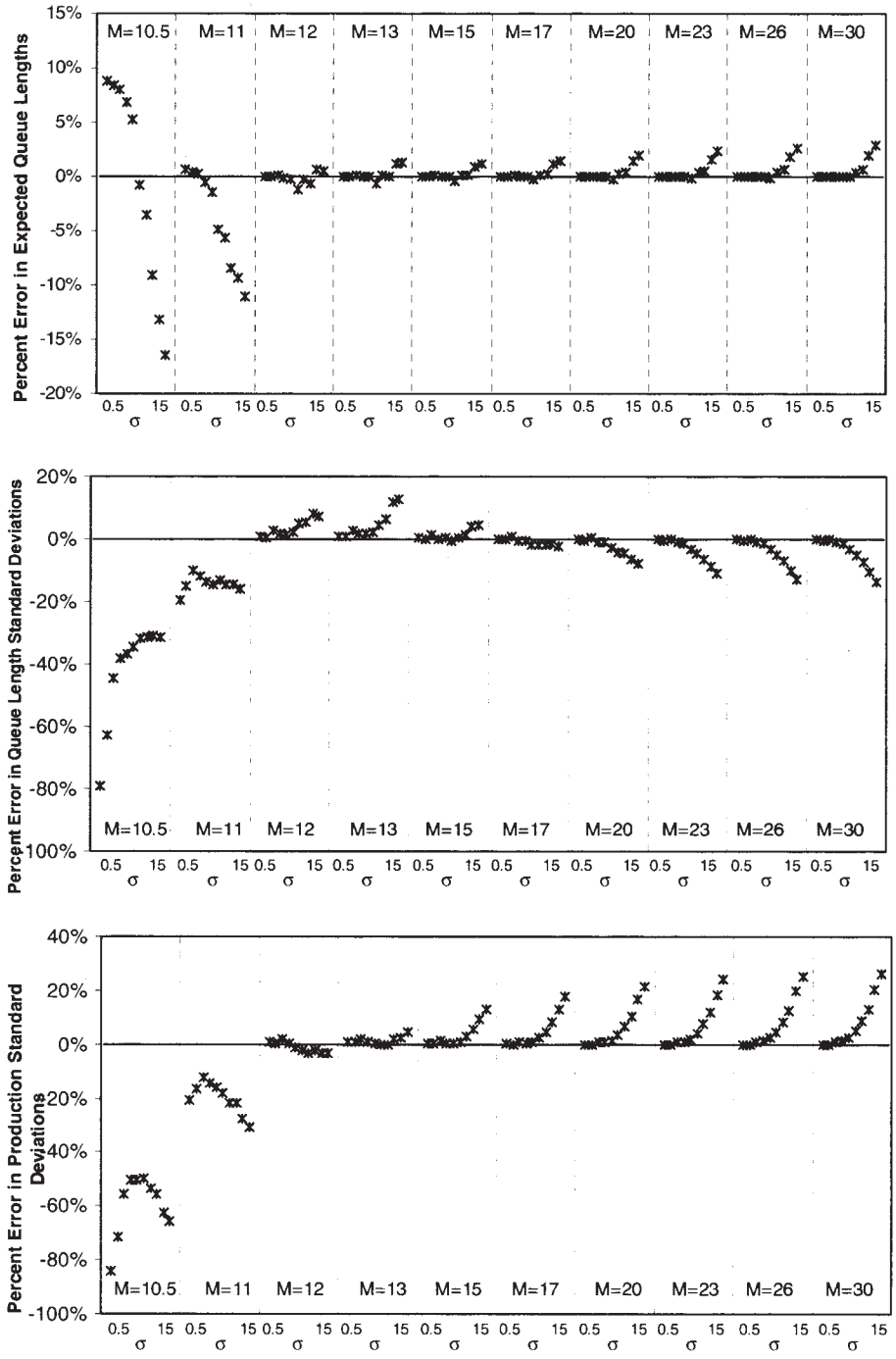

Figure 2. Approximated and simulated results for a 1-station model.

of 10 maximum service rates, M, ranging from 105% to 300% of the arrival rate (10.5 to 30). We set the clearing function parameter $\beta = M$ for all tests.

Figure 2 compares the model's estimates to those produced by simulations of the station across the 100 test scenarios. For each scenario, 200 1500 period simulations were performed, requiring more than 15 minutes of computing time on a SPARCstation Classic. (The SPARCstation calculated the estimated results almost instantly.) The first panel shows the percent difference between the estimated and simulated expected queue lengths; the second panel shows the percent difference between the estimated and simulated queue length standard deviations; and the third panel shows the percent difference between the esti-

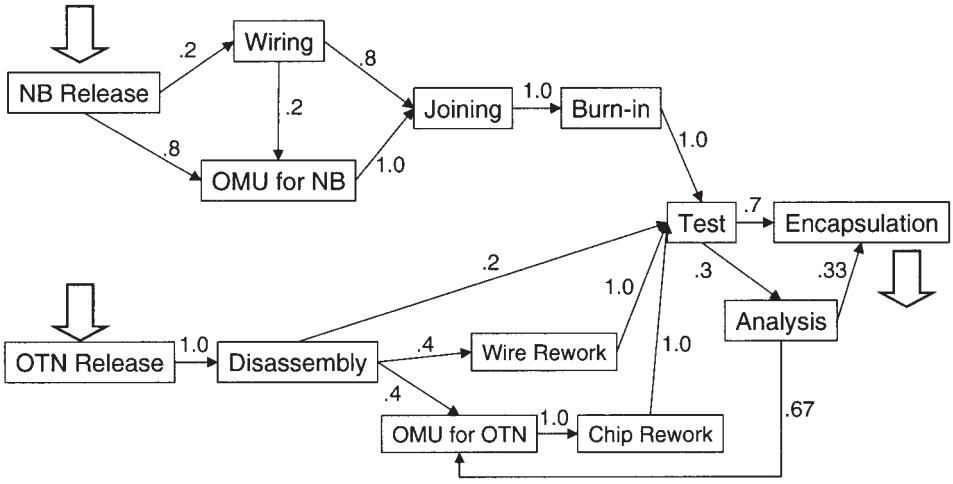

Figure 3. A job shop that manufactures subcomponents of mainframe computers.

mated and simulated production standard deviations. Each panel contains 10 subsidiary charts, with each chart graphing percent difference by input standard deviation, keeping the maximum service rate fixed. On the charts a negative percent difference corresponds to the estimate being less than the simulated value, and a positive percent difference corresponds to the estimate being greater than the simulated value.

Along the same guidelines discussed in Section 2, expected queue lengths were within 4%, and variance estimates were within 15%, of the simulation results, provided that the expected arrival rate was less than 5/6 of the asymptotic maximum output of a station (e.g., M at least 12), and the coefficient of variation of the input streams was less than 1 (e.g., input standard deviation of 10 or less). Past these bounds the approximations become inaccurate. The estimate of the expected queue length remains useful, never different by more than 20% from the simulated result; however, the estimates of the standard deviations can become useless. The approximations are significantly more sensitive to maximum service rates close to the expected input than they are to high input standard deviations.

Example 2: A Job Shop That Manufactures Subcomponents for Mainframe Computers. The following model refers to a manufacturing plant rather than an information flow in a distributed computing network, but is similar to a complex, multistage information flow. Fine and Graves [9] first considered this job shop as an application of the original TPM. Here, we consider a variant of the shop that uses the clearing functions as production functions.

Figure 3 diagrams the job shop. The shop contains 13 physical workstations; work enters through the NB Release and OTN Release stations and exits through the Encapsulation station. A single, independent computing cell supports each workstation. There is a feedback loop in the flow, related to subcomponents failing tests and requiring repair work. The numbers on each arc (i, j) represent the expected amount of work that arrives at station j given one unit of work at station i. It is assumed that the job shop perfectly follows model dynamics; all stations operate in discrete time, and each period's production is the amount specified by a clearing function.

Table 1 shows the input data for this idealized job shop, including the average work that arrives to the NB Release and OTN Release stations; the input standard deviations and expected production quantities at each station; and the maximum capacities of the cells supporting each station. We set both M_{cell} and β_{cell} for each cell to be the maximum capacity in Table 1.

Table 2 compares the analytic model's approximations to a 250,000 period simulation of the ideal job shop that employs the model's recursion equations directly. The work arrivals were modeled using Gamma distributions. Calculation of the analytic model's approximations was almost instantaneous on a SPARCstation classic; the 250,000 period simulation required over a half-hour on the same machine. The table compares the simulated and approximated expected queue lengths, queue length standard deviations, and production standard deviations at each station. Values for which the approximated and simulated moments differ by more than 10% are shown in boldface.

In general, the approximations are close to the simulated results. For expected queue lengths, no difference is greater than 3%. The production and queue length standard deviations also track fairly well with the simulated results; most differences were less than 10%. The few large differences appear to be explainable. The biggest differences were for the heaviest-loaded stations, Burn-In and Encapsulation,

Table 1. Input data for a mainframe subcomponents job shop.

Station	Average arrivals per period	Input standard deviation	Maximum cell capacity	Expected production
NB release	40	20	80	40
Wiring		6.3	16.3	12
OMU for NB		0	60	30.4
Joining		0	50	40
Burn-in		10	50	40
OTN release	30	12.5	60	30
Disassembly		12.5	45	30
OMU for OTN		0	60	29.6
Chip rework		12.5	45	29.6
Wire rework		12.5	18.8	12
Test		17.5	125	87.6
Analysis		0	50	26.3
Ecapsulation		7.5	87.5	70

and the station with the largest input standard deviation as a percentage of expected production, Wire Rework.

Example 3: Minimizing the Cost of Performance in the Job Shop. Suppose that each unit of capacity in the manufacturing subcomponents job shop costs \$100; thus, for example, the current M_{cell} of 80 at station NB Release costs \$8000. Hollywood [17] applies Little's Law to find weights that, when multiplied by the expected queue lengths, yield the total average waiting time to produce a mainframe subcomponent. (This is a weighted sum of the expected waiting times at each workstation, where the weights are the percentages of all subcomponent assemblies processed at each station.) We want the total average waiting time to remain at most 24.8421 periods. We directly apply the nonlinear programming methodology described in Section 5, and find that it is possible to reduce the total cost of capacity from \$74,640 to \$67,619 while maintaining the same performance. Figure 4 compares the original and approximately optimal capacity allocations at each station. As discussed, a SPARCstation classic found the solution in well under a minute of computing time.

The optimal solution economizes on capacity at a number of stations, especially NB release, OMU for NB, OTN Release, Analysis, and OMU for OTN. The optimal solution makes a few modest capacity increases at the Wiring, Joining, Burn-In, and Encapsulation stations, but these are significantly less than the capacity decreases. The total reduction in cost from the current to optimal solution is about 9.5%.

We solve the minimum-cost problem for a variety of constraints on total average waiting times, ranging from 9 periods to 50 periods. (The asymptotic minimum is 6.82 periods, which occurs when each station completes its entire queue each period.) Figure 5 presents the resulting cost-performance tradeoff curve.

Initially, one can greatly economize by accepting slightly worse network performance; one can more than halve the

Table 2. Approximated and simulated moments for the job shop.

	Expected queue length		Standard deviation of queue length		Standard deviation of production	
Station	Approximate	Simulated	Approximate	Simulated	Approximate	Simulated
NB release	88.9	89.1	39.6	40.0	9.4	9.3
Wiring	55.7	55.4	29.4	27.9	1.7	1.8
OMU for NB	64.9	64.9	19.9	19.9	4.8	4.8
Joining	224.0	224.3	82.9	81.2	3.1	3.1
Burn-in	217.1	221.0	**64.8**	**72.8**	**2.6**	**2.9**
OTN release	63.0	62.9	19.3	19.4	4.7	4.7
Disassembly	99.5	99.3	38.3	38.3	4.0	3.9
OMU for OTN	58.9	58.9	7.1	7.6	1.9	2.0
Chip rework	93.7	93.4	32.2	32.8	3.6	3.5
Wire rework	47.2	46.6	34.3	36.3	**3.5**	**3.0**
Test	304.3	304.5	68.2	73.4	6.2	6.5
Analysis	56.0	55.9	7.2	8.0	1.7	1.8
Encapsulation	365.8	371.3	**81.8**	**104.9**	**3.3**	**4.0**

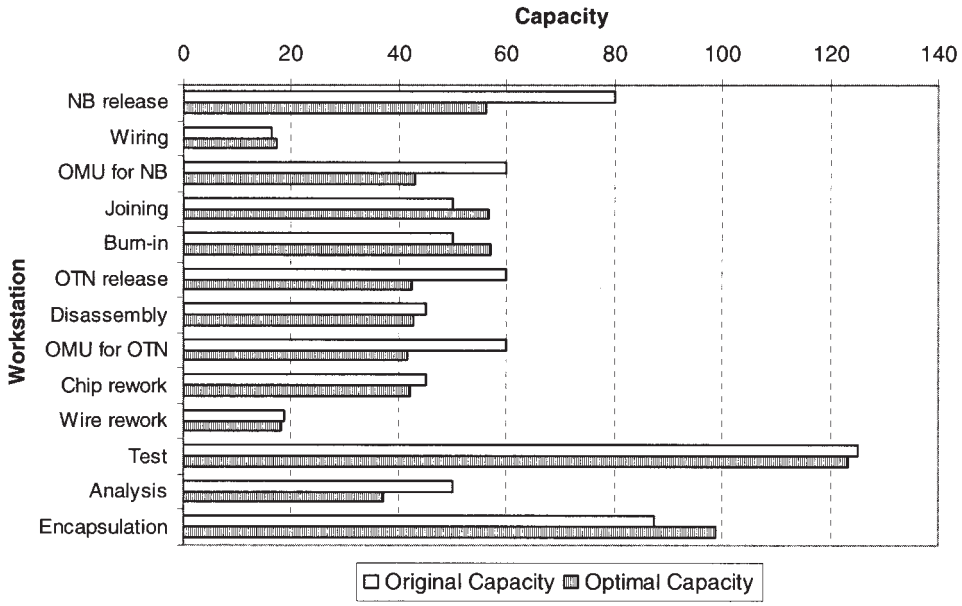

Figure 4. Current vs. optimal capacities in the mainframe subcomponents job shop.

capacity cost by increasing the total average waiting time from 9 periods to 13 periods. However, these savings level off quickly; it is not possible to halve the cost again, even by almost quadrupling the total average waiting time to 50 periods.

7. CONCLUSION

We have developed an approximate planning model that estimates the first two steady-state production and queue length moments for each processing station in a distributed network. We have seen how these approximations are generally reasonable provided that the network is well behaved (neither station loading nor arrival fluctuations are very high). Thus, the model can be used for a variety of high-level analytic purposes, including what-if analysis and examining tradeoffs between capacity placements. We also described the use of the model for a variety of approximate optimization applications, including minimum weighted

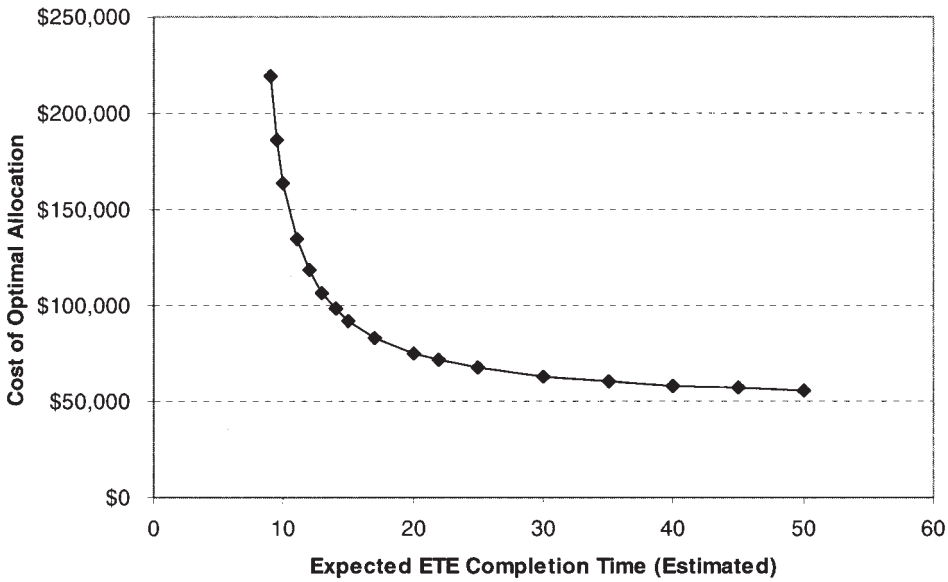

Figure 5. Optimal cost curve for the GLS-MR network.

queue lengths (and minimum weighted waiting times), and minimum cost.

We are interested in comparing the model in this paper with conventional queuing models. In general, we expect that conventional queuing models (cf. Jackson [18, 19], Baskett et al. [2], Kelly [22]) will be preferred when the shop has stations with independent service times for individual jobs (especially times close to exponential), operates continuously, and has jobs consisting of distinct classes of "customers" moving randomly around the shop according to priority rules. The model in this paper will be preferred when the job shop has queue-dependent processing rates, operates either in discrete periods or has discrete control-review periods, and is well defined by work flows or streams of jobs rather than by individual customers.

We conclude with several possibilities for extensions to the model, some of which have been researched. First, the model given in this paper assumes that one can find the inverse production functions and their derivatives easily. If this is not the case, it is possible to develop a model that uses only production functions; the development is similar to that of this paper's model, starting from the work-in-queue recursion equation (6) rather than the production recursion equation (7). The resulting model, however requires iterating between successive approximations of E(**Q**) and var(**Q**), starting from the first-order approximation of E(**Q**), to converge on a second-order approximation for the former and a first-order approximation of the latter. Details, and performance comparisons between the two models, may be found in Hollywood [17]; their accuracies are roughly equivalent.

Next, we might further explore models in which the control rules are sophisticated nonlinear functions of the work-in-queue levels at multiple stations, beyond the simple multi-task functions in this paper. We might consider nonlinear analogues of the Proportional Control rules of Denardo and Lee [5, 6], and Denardo and Tang [7], as well as analogues of the Constant Work-in-Process rules advocated by Spearman, Hopp, and Woodruff [24].

Third, we might be interested in developing continuous analogues of the discrete-period model in this paper. One possible approach might be to develop a model that has Brownian motion arrivals; another might be to consider the behavior of the model as the period length approaches zero.

Fourth, we might wish to explore the transitive behavior of the model given changing work arrival patterns. At first glance, transitive modeling appears to be a straightforward application of the models' recursion equations, calculating the moments of the queue lengths and completed over time, rather than the steady-state values. The challenge, however, is that the equations for calculating the moments are approximate, and that successive iterations of the recursion equations might become inaccurate quickly.

Finally, we may wish to explore differences between the approximated moments and the true moments. Hollywood [17] empirically found, for both single-station and multistation models, that the approximate moments tend to match the true moments well provided that the job shop stations are "well-behaved," in terms of neither being heavily-loaded nor facing large input fluctuations. However, we yet to find mathematical conditions that govern the relationship between the approximated moments and the true moments.

ACKNOWLEDGMENTS

The research in this paper was supported by the Department of Defense. I would like to thank Stephen Graves, Abraham J. Siegel Professor of Management and Engineering Systems at the Massachusetts Institute of Technology, for his assistance in reviewing this paper.

REFERENCES

[1] K. Baker, "Requirements planning," Logistics of production and inventory, Handbook in Operations Research and Management Science, Vol. 4, S. Graves, A.H.G. Rinnooy Kan, and P. Zipkin (Editors), North Holland, Amsterdam, 1993, pp. 571–627.

[2] F. Baskett, K.M. Chandy, R.R. Muntz, and F. Palacios, Open, closed and mixed networks of queues with different classes of customers, J ACM 22 (1975), 248–260.

[3] D.P. Bertsekas, Nonlinear programming, Athena Scientific, Belmont, MA, 1995.

[4] G.R. Bitran and D. Tirupati, "Hierarchical planning," Logistics of production and inventory, Handbook in Operations Research and Management Science, Vol. 4, S. Graves, A.H.G. Rinnooy Kan, and P. Zipkin (Editors), North Holland, Amsterdam, 1993, pp. 523–568.

[5] E.V. Denardo and Y.S. Lee, Pulling a Markov production system: I and II, Working Paper, Department of Operations Research, Yale University, New Haven, CT, 1987.

[6] E.V. Denardo and Y.S. Lee, Linear control of a Markov production system, Oper Res 40 (1992), 259–278.

[7] E.V. Denardo and C.S. Tang, Control of a stochastic production system with estimated parameters, Management Sci 43 (1997), 1296–1307.

[8] A.W. Drake, Fundamentals of applied probability theory, McGraw-Hill, New York, 1967.

[9] C.H. Fine and S.C. Graves, A tactical planning model for manufacturing subcomponents of mainframe computers, J Manuf Oper Management 2 (1989), 4–34.

[10] I. Foster, C. Kesselman, and S. Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, Int J Supercomput Appl 15(3) (2001), 200–222.

[11] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, Condor-G: A computation management agent for multi-institutional grids, Proc Tenth Int Symp High Performance Distributed Comput (HPDC-10), IEEE Press, New York, August 2001, pp. 55–63.

[12] R.G. Gallager, Discrete stochastic processes, Kluwer Academic, Boston, 1996.

[13] S.C. Graves, A tactical planning model for a job shop, Oper Res 34 (1986), 522–533.
[14] S.C. Graves, Determining the spares and staffing levels for a repair depot, J Manuf Oper Management 1 (1988), 227–241.
[15] A.S. Grimshaw, W.A. Wulf, Legion Team, The legion vision of a worldwide virtual computer, Commun ACM 40(1) (1997), 39–45.
[16] A.C. Hax, "Aggregate production planning," Handbook of operations research, Vol. 2, J. Moder and S.E. Elmaghraby (Editors), Von Nostrand, Reinhold, New York, 1978.
[17] J.S. Hollywood, Performance evaluations and optimization models for processing networks with queue-dependent production quantities, Ph.D. dissertation, Massachusetts Institute of Technology Operations Research Center, Cambridge, MA, June 2000.
[18] J.R. Jackson, Networks of waiting lines, Oper Res 5 (1957), 518–521.
[19] J.R. Jackson, Jobshop-like queueing systems, Management Sci 10 (1963), 131–142.
[20] U.S. Karmarkar, Capacity loading and release planning with work-in-progress (WIP) and leadtimes, J Manuf Oper Management 2 (1989), 105–123.
[21] U.S. Karmarkar, "Manufacturing lead times, order release, and capacity loading," Logistics of production and inventory, Handbook in Operations Research and Management Science, Vol. 4, S. Graves, A.H.G. Rinnooy Kan, and P. Zipkin (Editors), North Holland, Amsterdam, 1993, pp. 287–329.
[22] F.P. Kelly, Networks of queues with customers of different types, J Appl Probab 12 (1975), 542–554.
[23] J.A. Rice, Mathematical statistics and data analysis, 2nd edition, Duxbury Press, Belmont, MA, 1995.
[24] M.L. Spearman, W. Hopp, and D. Woodruff, A hierarchical control architecture for constant work-in-progress (CONWIP) production systems, J Manuf Oper Management 21 (1989), 147–171.
[25] Standard Performance Evaluation Corporation, SPEC CPU 2000 Results, 2004, http://www.spec.org/cpu2000/results/.
[26] M. Welsh, D. Culler, and E. Brewer, SEDA: An architecture for well-conditioned, Scalable Internet Services, Eighteenth Symp Oper Syst Principles (SOSP-18), Chateau Lake Louise, Canada, 21 October 2001, http://www.cs.berkeley.edu/~mdw/papers/seda-sosp01.pdf.