Heavy traffic resource pooling in parallel-server systems

J. Michael Harrison [a] and Marcel J. López [b]

[a] *Graduate School of Business, Stanford University, Stanford, CA 94305, USA*
E-mail: harrison_michael@gsb.stanford.edu
[b] *Graduate School of International Relations and Pacific Studies, University of California, San Diego, La Jolla, CA 92093-0519, USA*
E-mail: martylopez@ucsd.edu

Received 26 January 1999; revised 22 April 1999

We consider a queueing system with r non-identical servers working in parallel, exogenous arrivals into m different job classes, and linear holding costs for each class. Each arrival requires a single service, which may be provided by any of several different servers in our general formulation; the service time distribution depends on both the job class being processed and the server selected. The system manager seeks to minimize holding costs by dynamically scheduling waiting jobs onto available servers. A linear program involving only first-moment data (average arrival rates and mean service times) is used to define heavy traffic for a system of this form, and also to articulate a condition of overlapping server capabilities which leads to resource pooling in the heavy traffic limit. Assuming that the latter condition holds, we rescale time and state space in standard fashion, then identify a Brownian control problem that is the formal heavy traffic limit of our rescaled scheduling problem. Because of the assumed overlap in server capabilities, the limiting Brownian control problem is effectively one-dimensional, and it admits a pathwise optimal solution. That is, in the limiting Brownian control problem the multiple servers of our original model merge to form a single pool of service capacity, and there exists a dynamic control policy which minimizes cumulative cost incurred up to any time t with probability one. Interpreted in our original problem context, the Brownian solution suggests the following: virtually all backlogged work should be held in one particular job class, and all servers can and should be productively employed except when the total backlog is small. It is conjectured that such ideal system behavior can be approached using a family of relatively simple scheduling policies related to the $c\mu$ rule.

Keywords: heavy traffic, parallel-server systems, Brownian control problem, resource pooling

1. Introduction

This paper is concerned with dynamic scheduling of stochastic processing systems that have the following general structure (see figure 1). There are job classes indexed by $i = 1, \ldots, m$ and servers indexed by $k = 1, \ldots, r$. Jobs of each class arrive from outside the system and require a single service before they depart. Several different servers may be capable of processing any given job class, and the service

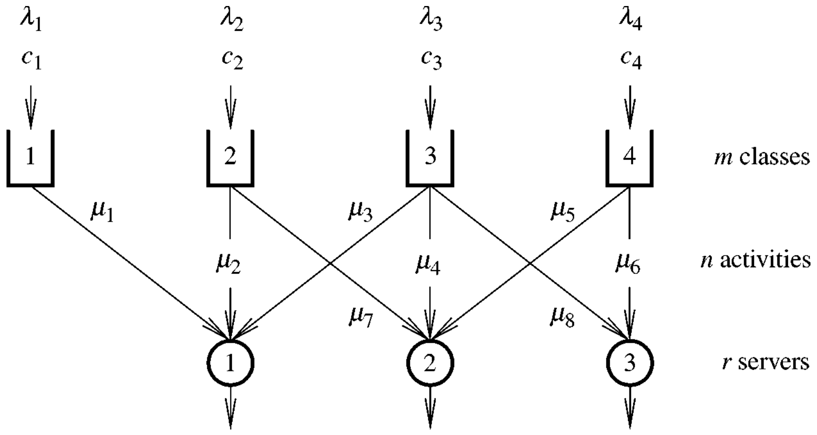

Figure 1. Example of a parallel-server system.

time distribution depends on both the class being processed and the server selected. (Here and later, we speak in terms of processing jobs rather than serving customers, but the terms "server" and "service time" are used in preference to "processor" and "processing time".) Thus there are $n \leqslant mr$ processing *activities* available to the system manager, each of which consists of a particular server processing a particular job class. For each activity $j = 1, \dots, n$ we denote by $i(j)$ the job class processed in that activity, and by $k(j)$ the server that does the processing. Also, let $j(i, k)$ be the activity in which server k processes class i (if there is one). Models having the structure described in this paragraph will be referred to hereafter as *parallel-server systems*.

For concreteness, we assume that the system is initially empty, and that services must be completed without interruption once they have been started, but neither of those assumptions is really important for the results ultimately obtained. It is convenient to imagine that jobs reside in buffers dedicated to their respective classes until they are ready to be processed (buffers are represented by open-ended rectangles in figure 1). The system manager need not specify which server will process a given job until the processing is to begin. Finally, holding costs are continuously incurred at a rate of c_i dollars per time unit for each class i job in the system, either waiting or being served. For future purposes, let us denote by λ_i the average arrival rate for class i jobs ($i = 1, \dots, m$) and by μ_j the reciprocal of the mean service time for activity j ($j = 1, \dots, n$). In the usual way, μ_j will often be referred to as the "mean service rate" for activity j.

Roughly speaking, the scheduling problem confronted by the system manager is one of dynamically allocating jobs to servers so that holding costs are minimized. In the interest of concreteness, readers may think in terms of minimizing long-run average cost per time unit, or of minimizing expected cost incurred up to a finite but distant time horizon, but for reasons explained below, our analysis is actually applicable under any optimality criterion that involves costs accumulated over relatively long time spans. At the mechanical level, a scheduling policy must specify, each time a server completes the processing of a job, which class that server will work on next. Also, each time a job arrives and finds one or more of its servers idle, the scheduling policy must specify which server will process that job. Of course, the scheduling policy must be non-anticipating in the sense that each decision is based solely on information available at the decision point. In the language of queueing network theory [5,12], the system manager's problem involves both *routing* decisions and *sequencing* decisions. That is, decisions must be made as to which job classes will be directed to which servers, and as to the order in which servers will process jobs from the various buffers. It is the opportunity for dynamic alternate routing that gives the problem its distinctive flavor.

Even in the simplest case with independent Poisson arrival processes, exponentially distributed service times for all activities, and a long-run average cost criterion, the dynamic scheduling problem described above defies exact solution. This is a typical difficulty in the study of stochastic processing networks, and for that reason, authors often resort to asymptotic analysis, searching for scheduling policies that are *asymptotically optimal* in the heavy traffic limit. Because routing decisions are endogenous in our parallel-server model, it is not immediately clear what is meant by "heavy traffic" in this context, but for that and other purposes we use a general approach developed by Harrison and Van Mieghem [8,9] based on earlier work of Laws [16]. In their approach one solves a linear program (LP) called the *static allocation problem* to determine a traffic intensity parameter for the processing system. When that parameter is near one, meaning that the load on some set of servers is approximately equal to their collective capacity, then we say that the system is in heavy traffic.

A general program for developing asymptotically optimal scheduling policies is laid out in [5]. That program, which has been successful in many contexts [2,10–12,15–20], can be described broadly as follows. First, following a procedure that is now more or less standard, one derives a *limiting Brownian control problem* that plausibly approximates the original dynamic scheduling problem after appropriate scaling. In our case that is just a matter of specializing to parallel-server systems more general derivations described in [8,16]. The limiting Brownian control problem is set in the context of a "limiting Brownian system model"; hereafter, those two terms will be used interchangeably, and the word "limiting" will often be dropped to reduce verbiage.

The second step in heavy traffic analysis is to solve the Brownian control problem, either in explicit mathematical terms or by numerical methods. Because fine structure is suppressed in the Brownian control problem, it is invariably simpler than the original problem which it approximates, but still, explicit mathematical solutions can only be expected in rather special circumstances. Third, one interprets the solution to the Brownian control problem in the context of the original problem. It is not always obvious how to do this, in which case some ingenuity must be applied to develop a proper implementation. Finally, closing the loop, one would like to prove rigorously that the proposed implementation is asymptotically optimal in the original problem, meaning that the percentage difference between its objective value and that under an optimal policy vanishes in the heavy traffic limit. These four steps, and their relationship to practical problem solving, will be explained more precisely in section 6 below, after the necessary mathematical framework has been established.

For parallel-server systems in heavy traffic, the second step described above may not have a favorable outcome. That is, the Brownian control problem may not be one which admits a simple solution. However, a simple solution *is* obtainable under an additional condition (that is, in addition to the assumption of heavy traffic) which we call complete resource pooling (or just *complete pooling*). In physical terms, this condition requires enough overlap in the capabilities of different servers so that they effectively form a single resource pool in the heavy traffic limit. In terms of the mathematical theory developed by Harrison and Van Mieghem [8,9] the complete pooling condition is expressed as follows: the Brownian control problem has an *equivalent workload formulation* in which the state of the system at any time t is given by a d-vector; we call d the effective system dimension, and say that complete pooling occurs when $d = 1$. The main result of this paper explains that condition in terms of the original model's structure, as follows: $d = 1$ if and only if all servers *communicate* in a certain sense (see section 2). That statement is proved as part of proposition 4, which provides three equivalent characterizations of complete pooling in parallel-server systems. Part of proposition 4 was actually proved by Laws [16] for a more general class of stochastic networks with alternate routing capabilities, but the further restriction to parallel-server systems allows us to characterize complete pooling in terms that are simple, explicit and intuitive. More will be said in section 8 about the relationship between our results and those of Laws [16].

This paper is restricted to heavy traffic analysis of parallel-server systems where complete pooling occurs, both for reasons of tractability and because complete pooling is a desirable design feature. (That is, one would always prefer to operate a processing system with enough overlap in server capabilities to enable complete pooling.) We find that the limiting Brownian control problem admits a *pathwise solution*; that is, because the multiple servers of our original model merge in the heavy traffic limit to form what is effectively a single pool of processing capacity, there exists a dynamic scheduling policy in the limiting model that minimizes the instantaneous cost rate at every point in time with probability one. That policy is obviously optimal in the limiting Brownian system model, regardless of what time horizon may be chosen and regardless of whether and how future costs may be discounted. At various points in this paper it will be called the "optimal Brownian control", the "pathwise Brownian solution", or to minimize the number of words involved, simply *the Brownian solution*.

Naively interpreted in terms of our original processing system, the Brownian solution has the following two properties. First, no server is ever idle unless the system is completely empty, and servers engage only in undominated *basic activities* (see section 2). Second, all buffers are empty at all times except the one corresponding to a lowest-ranking job class, where class rankings are determined by a generalization of the familiar $c\mu$ rule (see section 5). This ideal system behavior cannot be achieved in the original model, of course, but in future work we hope to show that such behavior can be approached using a simple type of discrete-review policy (see section 6).

The remainder of the paper is organized as follows. Section 2 introduces the static allocation problem as a means of defining heavy traffic, and defines several key concepts in terms of its optimal solution. This analysis depends heavily on prior work by Harrison and Van Mieghem [8,9] and we shall conform as closely as possible to the notation used in [8]. Section 3 spells out the probabilistic assumptions required in our analysis of parallel-server systems, and also introduces further notation. In section 4 we consider the Brownian control problem that is obtained as a formal heavy traffic limit, after suitable scaling, from a parallel-server system with traffic intensity parameter exactly equal to one. Building on general results developed earlier in [8,9], we identify the condition that is required for complete pooling in the Brownian control problem, expressing it in several equivalent forms (see proposition 4). Section 5 describes the pathwise Brownian solution that one obtains with complete pooling.

Section 6 defines the notion of asymptotic optimality that is relevant in our context, and introduces the family of discrete-review policies to be explored in future work. Those policies provide a potential means of approaching (as a heavy traffic limit) the ideal system behavior described by our Brownian solution, and we shall explain briefly how a system manager might make practical use of our theory.

Section 7 introduces the concept of super-server performance bounds, which are important in both the mathematical development and the interpretation of our theory. With regard to interpretation, one can make precise the idea of “heavy traffic resource pooling” by proving the asymptotic equivalence of a parallel-server system and a corresponding super-server model; the latter has a single processing resource, or super-server, whose capabilities combine in some sense the capabilities of all r servers in the original model. As readers will see in section 7, the super-server model that corresponds to a general parallel-server system is not as simple or straightforward as one might hope, and in particular, the best achievable performance in the relevant super-server model only bounds achievable performance in the original model in an asymptotic sense.

Finally, section 8 explains how our formal heavy traffic analysis is extended to parallel-server systems whose traffic intensity parameters are close to one but not necessarily equal to one. Section 8 also makes connections with the work of Laws [16] and with recent work by Kushner and Chen [13], which involves a different sort of heavy traffic limit for parallel-server systems.

At several points in the paper we use the following convention to distinguish between row vectors and column vectors: a notation like $x = (1, 1, 2)$, where the elements are separated by commas, means that x is a column vector, whereas a notation like $y = (1\ 1\ 2)$, where the elements are separated only by spaces, means that y is a row vector.

2. The static allocation problem

The general theory developed in [8] involves an $m \times n$ input–output matrix R and an $r \times n$ resource consumption matrix A that contain first-order data about processing activities. Recast in language that is natural for the parallel-server model considered here, the general descriptions of R and A are as follows: R_{ij} is the average rate at which activity j removes jobs from buffer i, and A_{kj} is the rate at which activity j consumes the capacity of server k. That is, in the current context one has $R_{i(j),j} = \mu_j$ and $R_{ij} = 0$ otherwise, while $A_{k(j),j} = 1$ and $A_{kj} = 0$ otherwise. For the example pictured in figure 1, this means that

$$R = \begin{bmatrix} \mu_1 & & & & & & & \\ & \mu_2 & & & & & \mu_7 & \\ & & \mu_3 & \mu_4 & & & & \mu_8 \\ & & & & \mu_5 & \mu_6 & & \end{bmatrix}$$

and

$$A = \begin{bmatrix} 1 & 1 & 1 & & & & & \\ & & & 1 & 1 & & 1 & \\ & & & & & 1 & & 1 \end{bmatrix}.$$

Together with the m-vector λ of mean arrival rates, these matrices comprise the data of the static allocation problem, which is now stated as follows: choose an n-dimensional column vector x and a scalar ρ so as to

$$\begin{aligned} &\text{minimize} && \rho && (2.1)\\ &\text{subject to} && Rx = \lambda, && (2.2)\\ & && Ax \leqslant \rho e, \quad \text{and} && (2.3)\\ & && x, \rho \geqslant 0, && (2.4) \end{aligned}$$

where e is an r-vector of ones. The n-vector x may be thought of as a *processing plan*, where x_j gives the long-run proportion of time that activity j is performed by its server, and ρ as the long-run utilization of the busiest server. Given these interpretations, our first constraint (2.2) merely enforces material conservation, while (2.3) requires that each server's utilization not exceed that of the busiest server. The objective is to minimize the maximum server utilization ρ, which motivates both an even distribution of work among servers and a restriction to the most efficient set of activities.

Assume throughout the remainder of this section that there is a unique optimal solution (x^*, ρ^*) for the static allocation problem. (This will be repeated as part of assumption 1 in section 4.) We shall refer to x^* as the *nominal processing plan*: the activities j for which $x_j^* > 0$ are called *basic activities*, and the remainder are called *nonbasic activities*. Let b denote the number of basic activities. Without loss of generality, assume that activities indexed by $j = 1, \ldots, b$ are the basic ones. For future purposes it will be convenient to write

$$R = [H \quad J] \quad \text{and} \quad A = [B \quad N], \tag{2.5}$$

where H and B each have b columns, corresponding to basic activities.

The static allocation problem (2.1)–(2.4) can be converted to equality form by adding non-negative slack variables $s_1, \dots, s_r$ in the constraints (2.3). When that is done, it is easy to verify that the $m + r$ rows of the resulting coefficient matrix are linearly independent, so a basis for the static allocation problem consists of $m + r$ variables. The unique optimal solution (x^*, ρ^*) is necessarily a basic solution (that is, an extreme point of the constraint set), and obviously $\rho^* > 0$, so there cannot be more than $m + r - 1$ basic activities. That is,

$$b \leqslant m + r - 1. \tag{2.6}$$

As it happens, attention will be concentrated here on the case $b = m+r-1$ (see below), where the optimal solution of the static allocation problem is necessarily nondegenerate. However, the degenerate case involves a potential terminological confusion that should be noted: it could then happen that a variable x_j which we call "nonbasic" is actually included in an optimal basis for the LP, but with value $x_j^* = 0$.

For the example pictured in figure 1, let us consider the specific numerical data

$$\lambda = \left(\tfrac{1}{2}, 1, 2, 3\right) \quad \text{and} \quad \mu = \left(\tfrac{5}{4}, \tfrac{5}{2}, 1, 2, 5, \tfrac{5}{2}, 4, \tfrac{4}{5}\right). \tag{2.7}$$

With these data, the unique solution of the static allocation problem (2.1)–(2.4) is

$$x^* = \left(\tfrac{2}{5}, \tfrac{2}{5}, \tfrac{1}{5}, \tfrac{9}{10}, \tfrac{1}{10}, 1, 0, 0\right) \quad \text{and} \quad \rho^* = 1. \tag{2.8}$$

Readers will note that all three constraints in (2.3) are binding. In other words, by using activities 1 through 6 in the indicated proportions, the system manager is just able, keeping all three servers fully occupied, to process jobs of the various classes at the required average rates. Any other feasible processing plan, including those that use activities 7 or 8, will necessarily exceed the capacity of at least one server. Thus, in order to avoid precipitous jobcount buildups, a system manager would like not only to keep all servers working but also to engage the servers as much as possible in basic activities. Given those restrictions, and remembering that the holding cost rate may be different for different classes, it is then natural to ask the following question. Can server work assignments be dynamically adjusted, relative to the nominal processing plan x^* in (2.8), so as to increase some jobcounts but decrease others in an economically favorable way?

For example, suppose that there is a large initial backlog in buffer 4 but moderate backlogs in buffers 1 through 3. (Readers may find it helpful to look again at figure 1.) Further suppose it is deemed desirable to shift the large backlog to buffer 1, perhaps because it has low holding cost, leaving moderate jobcounts in buffers 2 through 4. This can be accomplished by having server 2 spend more time on activity 5 (this drains buffer 4) and less time on activity 4 than indicated in the nominal plan, while server 1 spends more time on activity 3 (this compensates for the shift of server 2 capacity away from buffer 3) and less time on activity 1 (this will cause the content of buffer 1 to increase). Thus, even though server 1 cannot serve buffer 4 directly, all three servers must redistribute effort over the basic activities for which they are responsible in order to effect the desired transfer. Of course, the transfer would not be possible at all if the sets of classes processed by the various servers under our nominal plan did not overlap to an adequate degree. Such considerations motivate the following definition.

Definition (Communicating servers). Server k *communicates directly* with server k' if there exist basic activities j and j' such that $j = j(i, k)$ and $j' = j(i, k')$ for some class i. Server k *communicates* with server k' if there exist servers $k_1, \ldots, k_\omega$ such that $k_1 = k$, $k_\omega = k'$, and k_α communicates directly with $k_{\alpha+1}$ for all $\alpha = 1, \ldots, \omega - 1$.

In our example, servers 1 and 2 communicate directly, and so do servers 2 and 3. Thus each server communicates with every other one. Communication is an equivalence relation, so the servers can be partitioned into disjoint sets in which each server communicates only with those servers in its set. There is only one such set in our example, because all three servers communicate.

Proposition 1. If all servers communicate, then $b = m + r - 1$.

Proof. Consider a graph in which nodes represent either servers or job classes, and arcs represent basic activities. Associating activity j with arc j, server k with node k, and job class i with node $r + i$, we connect nodes $k(j)$ and $r + i(j)$ by arc j ($j = 1, \ldots, b$). This graph contains $m + r$ nodes, and the minimum number of arcs needed to have a connected graph is $m + r - 1$. Hence, if $b < m + r - 1$, then the graph consists of at least two connected subgraphs that are not connected to one another. Since each of these subgraphs contains at least one server node (because otherwise (2.2) would be violated), then not all servers communicate. □

The dual of the static allocation problem is stated as follows: choose (y, z) so as to

$$\begin{aligned} \text{maximize} \quad & y\lambda, && (2.9)\\ \text{subject to} \quad & yR \leqslant zA, && (2.10)\\ & ze \leqslant 1, \quad \text{and} && (2.11)\\ & z \geqslant 0, && (2.12) \end{aligned}$$

where y and z are row vectors of dimension m and r, respectively. Because of the special structure of R and A for our parallel-server system, each constraint in (2.10) involves a single component of y and a single component of z,

$$y_{i(j)}\mu_j \leqslant z_{k(j)} \quad \text{for all } j = 1, \ldots, n. \tag{2.13}$$

General interpretations of the dual variables y_i and z_k have been offered in [8], where they are respectively denoted as μ_i and π_k. These interpretations will be recapitulated in section 7 below. For the example pictured in figure 1, with the numerical data provided in (2.7), the unique optimal dual solution is

$$y^* = \left(\tfrac{1}{5}\ \tfrac{1}{10}\ \tfrac{1}{4}\ \tfrac{1}{10}\right) \quad \text{and} \quad z^* = \left(\tfrac{1}{4}\ \tfrac{1}{2}\ \tfrac{1}{4}\right). \tag{2.14}$$

Proposition 2. Suppose that the static allocation problem has a unique primal solution (x^*, ρ^*) and that it satisfies $Ax^* = \rho^* e$. Then a potential dual solution $(y,\ z)$ is optimal if and only if it satisfies the following four conditions:

$$y_{i(j)}\mu_j = z_{k(j)} \quad \text{for all } j = 1, \ldots, b; \tag{2.15}$$

$$z_1 + \cdots + z_r = 1; \tag{2.16}$$

$$y_{i(j)}\mu_j \leqslant z_{k(j)} \quad \text{for all } j = b+1, \ldots, n; \quad \text{and} \tag{2.17}$$

$$z_1, \ldots, z_r \geqslant 0. \tag{2.18}$$

Moreover, there exists an optimal dual solution $(y^*,\ z^*)$ for which all of the inequalities in both (2.17) and (2.18) are strict.

Proof. Consider the primal problem (2.1)–(2.4) in equality form, with non-negative slack variables $s_1, \ldots, s_r$ added to its original decision variables $\rho, x_1, \ldots, x_n$. The condition $Ax^* = \rho^* e$ says that all slack variables have value zero in the optimal primal solution, which is here assumed unique. A potential dual solution (y, z) is optimal if and only if it is dual feasible and satisfies complementary slackness. Given the representation (2.13) of our original dual constraints (2.10), readers can easily verify that complementary slackness takes the form of conditions (2.15) and (2.16). The residual requirements for $(y,\ z)$ to be dual feasible are then (2.17) and (2.18). The last statement of the proposition is established by specializing a result on "strict complementary slackness" that appears in [1, Exercise 4.20(b), p. 192]. □

Proposition 3. Given the hypotheses of proposition 2, the following three conditions are equivalent:

(i) the dual linear program (2.9)–(2.12) has a unique optimal solution $(y^*,\ z^*)$;

(ii) $b = m + r - 1$;

(iii) all servers communicate with one another.

Corollary. Suppose that the equivalent conditions (i)–(iii) hold, in addition to the hypotheses of proposition 2. Then all variables in the optimal dual solution (y^*, z^*) are strictly positive, and they satisfy each of the inequalities in (2.17) strictly. Also, the basis consisting of $x_1, \ldots, x_b$ and ρ remains uniquely optimal in a neighborhood of λ, and one has $Ax^* = \rho^* e$ in that neighborhood.

Proof. Proposition 1 shows that (iii) implies (ii). To prove the converse, suppose that the servers do not all communicate with one another. Then the r servers are partitioned into $q \geqslant 2$ communicating classes and the m job classes are partitioned into q corresponding subsets such that only servers from communicating class ℓ serve job classes from subset ℓ under the nominal plan, and servers from that communicating class do not serve any other job classes under the nominal plan ($\ell = 1, \dots, q$). Let r_ℓ be the number of servers in the ℓth communicating class, let m_ℓ be the number of job classes in the ℓth subset, and let b_ℓ be the number of basic activities j such that $k(j)$ is a server in the ℓth communicating class. Generalizing the proof of proposition 1, it is not difficult to establish that

$$b_\ell = m_\ell + r_\ell - 1 \text{ for } \ell = 1, \dots, q. \tag{2.19}$$

(It is crucial to this argument that the static allocation problem has a unique optimal solution, by assumption, and that it is necessarily a basic solution.) Summing (2.19) over $\ell = 1, \dots, q$ gives $b = m + r - q < m + r - 1$, so (ii) and (iii) are equivalent.

If (ii) holds, then (2.15) and (2.16) together constitute $m+r$ linearly independent equations (their coefficients are the columns of a basis matrix) in $m + r$ variables, so they uniquely determine the optimal dual solution (y^*, z^*), establishing that (ii) implies (i). To complete the proof of proposition 3 we assume that $b < m + r - 1$, which is the negation of (ii), and prove that the dual linear program (2.9)–(2.12) has at least two solutions, which is the negation of (i). It will be convenient to convert the dual linear program (2.9)–(2.12) to equality form, adding non-negative slack variables in all $n + 1$ of the original inequality constraints (2.10)–(2.11); to make connection with standard LP theory, one can further require that *all* of the dual decision variables be non-negative, because the equality constraints (2.2) in our primal problem could obviously be restated as $\geqslant$ inequalities without affecting the optimal solution, and that change would add constraints $y \geqslant 0$ to the dual problem statement (2.9)–(2.12). When the dual problem is restated in equality form its coefficient matrix has full row rank $n + 1$. We know that the dual problem has at least one basic optimal solution, and a basic solution has no more than $n + 1$ positive variables, including slacks. Now consider the optimal dual solution (y^*, z^*) described in the last sentence of proposition 2. That solution has a positive slack variable associated with each of the $n-b$ constraints in (2.17), and because $z_1^*, \dots, z_r^*$ are strictly positive as well, it follows immediately from the complementary slackness condition (2.15) that $y_1^*, \dots, y_m^*$ are strictly positive. Thus, we have a total of $m+r+(n-b)$ positive variables, including slacks. If $b < m + r - 1$ then $m + r + (n - b) > n + 1$, so the optimal dual solution (y^*, z^*) is not a basic solution, implying that the dual problem has multiple optima.

This completes the proof of proposition 3. The first statement of the corollary is immediate from proposition 2, and the second statement is a direct consequence of primal and dual nondegeneracy: that is, our unique primal optimum (x^*, ρ^*) is a nondegenerate basic solution when $b = m+r-1$, so it remains feasible for sufficiently small perturbations of the right-hand side vector, and it is uniquely optimal under such perturbations because the corresponding basic dual solution satisfies all the inequalities in (2.17)–(2.18) strictly. □

3. The dynamic scheduling problem

The dynamic scheduling problem for a parallel-server system was explained verbally in section 1. Translating that problem description into precise mathematical terms is complicated and cumbersome, primarily because of the need to make precise the notion of non-anticipating policies. A complete mathematical description of the dynamic scheduling problem is not really necessary for our purposes, but to make connection with the general theory developed in [8] it *is* necessary to introduce some additional notation.

The probabilistic primitives in our model are a collection of independent, m-dimensional stochastic processes $F^0, F^1, \dots, F^n$ that all have time domain $[0, \infty)$. Let us denote by F_i^j the ith component process of F^j. One interprets $F_i^0(t)$ as the number of arrivals into buffer i over the time interval $[0, t]$. For each $j = 1, \dots, n$ and $i = 1, \dots, m$ one interprets $F_i^j(t)$ as the number of departures from buffer i resulting from the first t time units devoted to activity j by its server. Thus, $F_{i(j)}^j$ is the renewal process generated by the i.i.d. service time sequence for activity j, and $F_i^j = 0$ otherwise. Rather than imposing assumptions about the fine stochastic structure of the external arrival process F^0, we just assume directly that it satisfies a functional central limit theorem, as follows: there exists a strictly positive m-vector λ and an $m \times m$ covariance matrix Γ^0 such that the scaled process

$$\varepsilon\left[F^0\left(t/\varepsilon^2\right) - \lambda t/\varepsilon^2\right], \quad t \geqslant 0, \tag{3.1}$$

converges weakly as $\varepsilon \downarrow 0$ to a Brownian motion with zero drift and covariance matrix Γ^0. Because the service times for each processing activity are assumed to be i.i.d., and if we further assume that service times have finite second moments, the following is immediate from the functional central limit theorem for renewal processes: for each $j = 1, \dots, n$ let σ_j^2 be the variance of service times for activity j, let R^j be the jth column of the $m \times n$ matrix R defined in section 2, and let Γ^j be the $m \times m$ matrix having $\mu_j^3 \sigma_j^2$ as its $i(j)$th diagonal element and having all other elements equal to zero; then the scaled processes

$$\varepsilon\left[F^j\left(t/\varepsilon^2\right) - R^j t/\varepsilon^2\right], \quad t \geqslant 0, \tag{3.2}$$

converge weakly as $\varepsilon \downarrow 0$ to a Brownian motion with zero drift and covariance matrix Γ^j.

As in [5,8], one can describe a dynamic scheduling policy (or just *policy*, for short) for the parallel-server system by means of an n-dimensional stochastic process $T = \{T(t),\ t \geqslant 0\}$ with components T_j. One interprets $T_j(t)$ as the total time devoted to activity j by its server over the interval $[0, t]$, and hence the m-dimensional *jobcount process* may be defined as

$$Q(t) = F^0(t) - \sum_{j=1}^{n} F^j\left(T_j(t)\right), \quad t \geqslant 0. \tag{3.3}$$

(As in section 1, we have assumed the system to be initially empty in writing this key relationship. Our treatment can easily be extended to allow nonzero initial jobcounts, but doing so adds nothing conceptually and it complicates things slightly.) Of course, a policy T must satisfy a number of conditions (continuous, nondecreasing, null at time zero, and more) in order to be admissible, but there is no need to spell out those conditions here. Given a policy T with jobcount process Q defined by (3.3), and with c taken to be a row vector, the corresponding cost rate process $C = \{C(t),\ t \geqslant 0\}$ is given by

$$C(t) = cQ(t), \quad t \geqslant 0. \tag{3.4}$$

That is, the quantity $C(t)$ represents the instantaneous cost rate at time t under policy T.

Assuming again that the static allocation problem has a unique solution (x^*, ρ^*) and given an arbitrary policy T, let us define as in [8] the corresponding n-dimensional process $V = \{V(t),\ t \geqslant 0\}$ of *deviation controls*:

$$V(t) = x^*t - T(t), \quad t \geqslant 0. \tag{3.5}$$

Also as in [8], we set $p = r + n - b$ and define a $p \times n$ matrix K via

$$K = \begin{bmatrix} B & N \\ 0 & -I \end{bmatrix}, \tag{3.6}$$

where I is the $(n-b)\times(n-b)$ identity matrix, and then define a p-dimensional process

$$I(t) = KV(t), \quad t \geqslant 0. \tag{3.7}$$

Exactly as in [8], the first r components of the process I are interpreted as cumulative idleness processes for servers $1, \dots, r$, respectively, while the last $n - b$ components represent cumulative time allocations to nonbasic activities. Thus, all components of I must be nondecreasing if T is to be an admissible policy.

Finally, in formulating and analyzing the Brownian control problem that approximates our dynamic scheduling problem, we shall speak in terms of scaled processes $Y = \{Y(t),\ t \geqslant 0\}$, $Z = \{Z(t),\ t \geqslant 0\}$, $U = \{U(t),\ t \geqslant 0\}$ and $\zeta = \{\zeta(t),\ t \geqslant 0\}$ that are obtained from V, Q, I and C using a small parameter $\varepsilon > 0$ as follows:

$$Y(t) = \varepsilon V\big(t/\varepsilon^2\big), \tag{3.8}$$

$$Z(t) = \varepsilon Q\big(t/\varepsilon^2\big), \tag{3.9}$$

$$U(t) = \varepsilon I\big(t/\varepsilon^2\big), \quad \text{and} \tag{3.10}$$

$$\zeta(t) = \varepsilon C\big(t/\varepsilon^2\big) \tag{3.11}$$

for $t \geqslant 0$. In defining scaled processes by means of a small parameter ε, we are following the notational convention used in [8]. In [6] and most other work on heavy traffic limit theory, the scaling involves a large parameter n, but the two definitional systems are completely equivalent if one makes the association $\varepsilon = n^{-1/2}$. (In this paper, of course, the letter n is used with a different meaning.) The essential feature of the scaling embodied in (3.8)–(3.11) is that large amounts of “real time” or “original time" are compressed into moderate amounts of "scaled time". To be specific, one unit of scaled time is equivalent to ε^{-2} units of real time, so this scaling is only appropriate when time spans of order ε^{-2} are the relevant ones for purposes of performance measurement (see section 6 for an example). This focus on relatively long time spans is a central feature of "heavy traffic limit theory". As a complement to the time scale compression by a factor of ε^{-2}, we compress the spatial scales in (3.8)–(3.11) by a factor of ε^{-1} in order that quantities of interest remain moderate in absolute magnitude. As we shall explain in section 8, a meaningful Brownian approximation can only be obtained under such scaling if the traffic intensity parameter ρ^* is close to 1, where "closeness" is measured in terms of the scaling parameter ε.

Readers will note that all of the processes defined in (3.8)–(3.11) depend on the scaling parameter ε and the policy T with which one starts, but that dependence is not reflected in our notation. Because we are not concerned with rigorous limit theory at this stage of the development, it is not necessary to introduce more precise and elaborate notation.

4. The limiting Brownian control problem

This section focuses on the Brownian control problem that is obtained, following the general recipe laid out in [5,8], as a formal heavy traffic limit of our dynamic scheduling problem after appropriate scaling. Following a long-established tradition in heavy traffic analysis, we shall first discuss the case of a single system whose traffic intensity parameter, properly defined, is precisely equal to 1, and later generalize the analysis to a parametric family of models whose traffic intensity parameters approach 1 as a limit. To be more specific, we first consider a single system whose first-order data satisfy the following condition.

Assumption 1 (Heavy traffic). The data (R, A, λ) of the static allocation problem are such that

(i) its solution (x^*, ρ^*) is unique;

(ii) $\rho^* = 1$; and

(iii) $Ax^* = e$.

These are the same conditions used to define heavy traffic in [8], and two points made in the discussion there deserve at least a brief mention. First, the requirement $Ax^* = e$ seems to say that all servers are fully utilized under the nominal processing plan, but that may simply mean that analysis is restricted to the "bottleneck subnetwork" of critically loaded servers. That is, the condition $Ax^* = e$ may be interpreted to mean that servers which need not be fully utilized to handle the given load have simply been deleted from our model. Second, it may seem needlessly restrictive to require that the static allocation problem have a unique solution, but multiple optima lead to Brownian approximations of a more complicated form than we are prepared to treat.

Given a single system whose first-order data satisfy assumption 1, and assuming that the time horizon over which we seek to minimize cost is relatively long, suppose that the various processes associated with an arbitrary policy T are rescaled as in (3.8)–(3.11) using a small parameter $\varepsilon > 0$. In [5,8] an informal argument is given to suggest that as $\varepsilon \downarrow 0$, the scaled dynamic scheduling problem is increasingly well approximated by a simpler dynamic control problem involving an m-dimensional Brownian motion $X = \{X(t),\ t \geqslant 0\}$. To be more specific, X has zero drift, covariance matrix

$$\Gamma = \Gamma^0 + \sum_{j=1}^{b} x_j^* \Gamma^j, \tag{4.1}$$

and initial state $X(0) = 0$. In the limiting Brownian control problem, an admissible control takes the form of an n-dimensional process Y such that

$$Y \text{ is non-anticipating with respect to } X, \tag{4.2}$$

$$Z(t) \geqslant 0 \text{ for all } t \geqslant 0, \text{ and} \tag{4.3}$$

$$U \text{ is nondecreasing with } U(0) \geqslant 0, \text{ where} \tag{4.4}$$

$$Z(t) = X(t) + RY(t) \text{ for all } t \geqslant 0 \text{ and} \tag{4.5}$$

$$U(t) = KY(t) \text{ for all } t \geqslant 0. \tag{4.6}$$

In the obvious way, the processes Y, Z and U appearing in this problem statement represent limits as $\varepsilon \downarrow 0$ of scaled processes denoted by the same letters in section 3, and we associate with each admissible control Y a corresponding (scaled) cost rate process ζ via

$$\zeta(t) = cZ(t), \quad t \geqslant 0. \tag{4.7}$$

As explained in [8,9], the Brownian system model (4.2)–(4.6) has an "equivalent workload formulation" in which the state of the system at any time t is described by a "workload vector" $W(t)$ having dimension $d \leqslant m$; we shall occasionally refer to d as the *effective dimension* of the original Brownian system model (4.2)–(4.6). Relevant aspects of the equivalent workload formulation will be reviewed shortly. Our main result is the following.

Proposition 4 (Complete pooling). Consider the Brownian model (4.2)–(4.6) that is derived from a parallel-server system whose data (R, A, λ) satisfy assumption 1. Its equivalent workload formulation has dimension $d = 1$ if and only if (R, A, λ) further satisfy the equivalent conditions (i)–(iii) of proposition 3.

Proof. By strong duality, conditions (i) and (ii) of assumption 1 imply that the dual linear program (2.9)–(2.12) has at least one feasible solution $(y^\ell,\ z^\ell)$ such that $y^\ell \lambda = 1$. As in [8, section 2], let d be the dimension of the linear space spanned by such vectors y^ℓ. (Again readers are reminded that the dual variables denoted by y and z in this paper are denoted in [8] by μ and π, respectively.) The $d \times m$ matrix M is then defined as

$$M = \begin{bmatrix} y^1 \\ \vdots \\ y^d \end{bmatrix} \tag{4.8}$$

for any linearly independent $y^1, \dots, y^d$. It follows from [8, proposition 4] that $d = 1$ if and only if the dual linear program (2.9)–(2.12) has a unique optimal solution (y^*, z^*). Because assumption 1 includes the hypotheses of proposition 2, the desired conclusion is then immediate from proposition 3. □

To recapitulate, proposition 4 gives a necessary and sufficient condition, involving first-order system data (R, A, λ), for our limiting Brownian control problem to have effective dimension $d = 1$. Let us assume that this condition holds, and as before denote by (y^*, z^*) the unique optimal dual solution for the static allocation problem. Adopting the "canonical" representation of workload proposed in [8], the equivalent workload formulation of (4.2)–(4.6) involves the one-dimensional Brownian motion $\xi = \{\xi(t),\ t \geqslant 0\}$ defined by

$$\xi(t) = y^* X(t), \quad t \geqslant 0. \tag{4.9}$$

Also, for general d the equivalent workload formulation described in [8] involves a $d \times p$ matrix G, so in our case G is just a p-dimensional row vector (recall from section 3 that $p = r + n - b$), and its precise content will be specified below in (5.1)–(5.3). In the equivalent workload formulation one takes the point of view that a system manager directly chooses the processes Z and U appearing in (4.2)–(4.6), rather than choosing the vector Y of scaled deviation controls. To be more precise, an *admissible strategy* is defined as a pair (Z, U) that satisfies the following conditions:

$$U \text{ is non-anticipating with respect to } X, \tag{4.10}$$

$$Z(t) \geqslant 0 \text{ for all } t \geqslant 0, \tag{4.11}$$

$$U \text{ is nondecreasing with } U(0) \geqslant 0, \text{ and} \tag{4.12}$$

$$W(t) = \xi(t) + GU(t) \text{ for all } t \geqslant 0, \text{ where} \tag{4.13}$$

$$W(t) = y^* Z(t) \text{ for all } t \geqslant 0. \tag{4.14}$$

In this equivalent workload formulation the processes Z and U have the same interpretation as before: components of Z represent scaled jobcount processes, or scaled buffer content processes, for the various job classes; the first r components of U represent scaled cumulative idleness processes for the various servers; and the last $n - b$ components of U represent scaled cumulative time allocations to the various nonbasic activities. We associate with each admissible strategy (Z, U) a cumulative cost process ζ via (4.7). One interprets the constant y_i^* as the "workload contribution" of a class i job, and in (4.14) the (scaled) workload vector $W(t)$ is obtained by summing over job classes i the (scaled) workload contents $y_i^* Z_i(t)$ in their respective buffers.

As explained in [8, section 4], the corresponding interpretation of our optimal dual variable z_k^* is the following: it is the average rate at which server k drains workload from the system when exclusively engaged in basic activities. In these verbal interpretations of the optimal dual variables y_i^* and z_k^*, the word workload means the time required to empty all buffers in a fluid analog of our parallel-server system; see [8] for elaboration and details.

5. Pathwise solutions with complete pooling

Before introducing the pathwise solution of our Brownian control problem, it is necessary to describe precisely the process

$$L(t) = GU(t), \quad t \geqslant 0, \tag{5.1}$$

appearing on the right-hand side of (4.13). By combining the definition of G provided in [8, equation (3.11)] with (2.5), (3.5)–(3.8) and (3.10) above, readers can verify that

$$L(t) = \sum_{k=1}^{r} z_k^* \, U_k(t) + \sum_{\ell=1}^{n-b} g_{b+\ell}^* \, U_{r+\ell}(t), \tag{5.2}$$

where

$$g_j^* = z_{k(j)}^* - y_{i(j)}^* \mu_j \quad \text{for } j = b+1, \ldots, n. \tag{5.3}$$

(Recall that y_i and z_k are denoted in [8] as μ_i and π_k, respectively.) The corollary to proposition 3 says that all of the constants z_k^* and g_j^* appearing in (5.2) are strictly positive (that is, all components of the p-vector G are strictly positive), so (4.12) implies that L is nondecreasing with $L(0) \geqslant 0$ under any admissible strategy (Z, U).

A pathwise solution of (4.10)–(4.14) can now be constructed as follows. First, define the one-dimensional reflected or regulated Brownian motion W^* via

$$W^*(t) = \xi(t) + L^*(t), \quad t \geqslant 0, \tag{5.4}$$

where

$$L^*(t) = -\inf_{0 \leqslant s \leqslant t} \xi(s), \quad t \geqslant 0. \tag{5.5}$$

Then L^* is continuous and nondecreasing with $L^*(0) = 0$, and it is well known that

$$L^* \text{ increases only at times } t \text{ when } W^*(t) = 0, \tag{5.6}$$

cf. [4, chapter 2]. From those properties it follows that, for any admissible strategy (Z, U), the workload process W defined by (4.14) satisfies $W(t) \geqslant W^*(t)$ for all $t \geqslant 0$ (thus, W^* is a *pathwise bound* for W). Thus, recalling the definition (4.7) of the cost rate process ζ under an admissible strategy (Z, U), and further defining

$$\theta_i = \frac{c_i}{y_i^*} \quad \text{for } i = 1, \dots, m, \tag{5.7}$$

$$\theta^* = \min \{\theta_1, \dots, \theta_m\}, \tag{5.8}$$

and

$$\zeta^*(t) = \theta^* W^*(t), \quad t \geqslant 0, \tag{5.9}$$

one has the following:

$$\begin{aligned}\zeta(t) &= \sum_{i=1}^{m} c_i\, Z_i(t) = \sum_{i=1}^{m} \theta_i\, y_i^*\, Z_i(t)\\ &\geqslant \theta^* \sum_{i=1}^{m} y_i^*\, Z_i(t) = \theta^*\, W(t)\\ &\geqslant \theta^*\, W^*(t) = \zeta^*(t)\end{aligned} \tag{5.10}$$

for all $t \geqslant 0$. (Again, this is a pathwise relationship between two stochastic processes ζ and ζ^*.) To construct an admissible pair (Z^*, U^*) that actually achieves the pathwise bound ζ^*, let the job classes be numbered so that

$$\theta_1 \leqslant \cdots \leqslant \theta_m. \tag{5.11}$$

One way to achieve the bound (others are possible, as explained below) is by taking

$$Z_1^*(t) = \frac{1}{y_1^*} W^*(t), \tag{5.12}$$

$$Z_2^*(t) = \cdots = Z_m^*(t) = 0, \tag{5.13}$$

$$U_1^*(t) = \frac{1}{z_1^*} L^*(t), \quad \text{and} \tag{5.14}$$

$$U_2^*(t) = \cdots = U_p^*(t) = 0 \tag{5.15}$$

for $t \geqslant 0$. To establish the pathwise optimality of this pair (Z^*, U^*) readers must verify that it satisfies all of the admissibility conditions (4.10)–(4.14) and that $\zeta^* = cZ^*$, both of which are easy to do. Of course, one must use the fact that $\theta_1 = \theta^*$ by (5.11); if the first inequality in (5.11) is not strict, then one could substitute for class 1 in (5.12) any other class i such that $\theta_i = \theta^*$ and the same conclusion would be reached. In similar fashion, the choice of U_1^* to be the only nonzero component of U^* is completely arbitrary: given that all the constants z_k^* appearing in (5.2) are strictly positive, one could substitute for server 1 in (5.14) any other server $k \in \{1, \dots, r\}$. Furthermore, because all of the constants g_j^* in (5.2) are strictly positive as well, one could choose the nonzero component of U^* in 5.14 to be

$$U_{r+\ell}^*(t) = \frac{1}{g_{b+\ell}^*} L^*(t), \quad t \geqslant 0, \tag{5.16}$$

for one particular $\ell \in \{1, \ldots, n - b\}$ and then take all other components of U^* to be zero as in (5.15). Roughly speaking, this would be interpreted as follows. The system manager, seeing that workload is approaching zero and thus all servers cannot be kept efficiently employed, would choose not to idle any server, but rather to engage some server in a nonbasic activity. Because nonbasic activities effectively squander capacity, workload rises precipitously under Brownian scaling, and so shortly the system manager can return to a mode in which all servers are fully occupied with basic activities.

Having specified in (5.12)–(5.15) one particular policy (Z^*, U^*) as a candidate for optimality, we summarize the arguments above in the following proposition, a more general version of which was proved by Laws [16].

Proposition 5. Consider the equivalent workload formulation (4.10)–(4.14) of the Brownian control problem (4.2)–(4.6) that is derived from a parallel-server system whose data (R, A, λ) satisfy assumption 1. Further, suppose that those data satisfy the complete pooling condition articulated in propositions 3 and 4. Given the numbering convention (5.11), the pair (Z^*, U^*) defined by (5.4)–(5.5) and (5.12)–(5.15) has associated cost rate process $\theta^* Z^* = \zeta^*$, and thus it is pathwise optimal in the sense of (5.10).

For an interpretation of all this in terms of our original scheduling problem, or rather to begin development of an interpretation, we note the following. First, since $U_1, \ldots, U_r$ are scaled cumulative idleness processes for the r different servers and the remaining components of U represent scaled cumulative time allocations to nonbasic activities, (5.15) says that only server 1 will ever experience idleness, and that only basic activities will ever be undertaken. Second, (5.14) and (5.6) together say that server 1 will only experience idleness when the system is completely empty (that is, when the workload is zero). Finally, (5.11) and (5.12) together say that all backlogged work is to be held in the form of jobs from that class i which minimizes the index $\theta_i = c_i / y_i^*$. This index represents the holding cost incurred (per hour, say) per unit of work held in buffer i.

It is more or less obvious that no strategy can do better than one which both achieves the minimal workload process W^* and holds all work in that buffer for which the holding cost per unit of work is minimal; those are the definitive characteristics of (Z^*, U^*). It is equally obvious that such ideal system behavior, although it is achievable in the limiting Brownian model, can only be approached in our original parallel-server system. The question of *how* to approach that ideal behavior under heavy traffic conditions will be addressed briefly in the next section.

To show the connection between the classical $c\mu$ rule and our ranking (5.11) of the job classes, let us denote by J the set of all basic activities j for which server 1 is responsible, and by I the set of all job classes i processed in those activities. That is, J consists of those $j \in \{1, \ldots, b\}$ such that $k(j) = 1$, and I consists of those $i \in \{1, \ldots, m\}$ such that $i(j) = i$ for some $j \in J$. (The focus on server 1 is arbitrary.)

If i is the class processed by some particular activity $j \in J$, then we have from (2.15) that $y_i^* \mu_j = z_1$ and, hence, $\theta_i = c_i / y_i^* = c_i \mu_j / z_1$. That is, ranking classes $i \in I$ according to the index θ_i is precisely the same, from the perspective of server 1, as ranking those classes according to the $c\mu$ criterion. A similar statement holds for each of the other servers, of course.

6. Asymptotically optimal scheduling policies

In section 1 a four-step procedure was outlined for heavy traffic analysis of a dynamic scheduling problem. At this point the first two steps have been completed for the case of parallel-server systems, given that they satisfy our complete pooling condition: the associated Brownian control problem was written out in section 4, following the purely formal procedure proposed in [5,8], and a pathwise optimal solution of that Brownian control problem was derived in section 5. Step three is to interpret the Brownian solution in our original problem context, showing how it can be approached through a sequence of implementable scheduling policies. Step four is to prove that, in some appropriate sense, no sequence of admissible scheduling policies in the original model can "do better than" the Brownian solution, thus establishing "asymptotic optimality" of the policies identified in step three. Steps three and four together can be viewed as a confirmation that the purely formal argument used to justify our "limiting Brownian control problem" was in fact correct. That is, we seek to confirm that the Brownian solution does correspond to best achievable performance in our original model, in an appropriate asymptotic sense.

The last two steps of the heavy traffic program for parallel-server systems will not be undertaken in this paper, but some additional development is appropriate to set the stage for future work, and to facilitate interpretation of the pathwise optimal Brownian solution derived in section 5. Consider again the general dynamic scheduling problem described in section 3, recalling that a scheduling policy takes the form of a nondecreasing vector process $T = \{T(t),\ t \geqslant 0\}$ whose components represent cumulative time allocations to n different activities that may be undertaken by r different servers. Given a parallel-server system whose data (R, A, λ) satisfy both assumption 1 and our complete pooling condition, let T^ε be a family of admissible scheduling policies parameterized by $\varepsilon > 0$. For each policy in the family, define a scaled jobcount process Z^ε and scaled cost rate process ζ^ε via (3.3), (3.4), (3.9) and (3.11), using T^ε in place of T. We conjecture that

$$\limsup_{\varepsilon \downarrow 0} P\bigl\{\zeta^\varepsilon(t) \leqslant x\bigr\} \leqslant P\bigl\{\zeta^*(t) \leqslant x\bigr\} \tag{6.1}$$

for all $t > 0$ and all $x > 0$. This result, if it is true, says that the pathwise optimal Brownian cost rate process ζ^* asymptotically dominates, in the sense of stochastic ordering at every time t, the best achievable scaled cost rate in our original model. To repeat a point made earlier, the asymptotic mode of analysis under discussion here (as the scaling parameter ε becomes small) is one that considers ever longer spans of time, and that adjusts the spatial scales of all processes in corresponding fashion so that the quantities of interest remain moderate. The precise restrictions that define an "admissible policy" T in the parallel-server model have not been spelled out in this paper, but we conjecture that (6.1) remains valid even when an unrealistically large class of policies is considered, including policies that look into the future and policies that may split server effort among several activities simultaneously. More will be said in the next section about potential means of proving (6.1).

As a companion to the conjectured asymptotic bound (6.1), one would like to identify a family of implementable policies T^ε such that

$$\zeta^\varepsilon \Rightarrow \zeta^* \quad \text{as } \varepsilon \downarrow 0, \tag{6.2}$$

where $\Rightarrow$ signifies weak convergence (or convergence in distribution) in an appropriate function space setting. This would mean that, for arbitrary $t > 0$ and $x > 0$, the probability on the left side of (6.1) actually approaches the probability on the right side as $\varepsilon \downarrow 0$. Together, then, (6.1) and (6.2) would justify calling the identified sequence of policies *asymptotically pathwise optimal*.

A family of discrete-review policies that are asymptotically optimal in this sense was constructed in [7] for a simple two-server example (see section 7 below). In future work that approach will be extended to a more general class of parallel-server systems, and a brief description of the ideas involved will help readers to interpret the Brownian solution obtained in section 5. For concreteness, consider the three-server model described in figure 1; additional numerical data are provided in (2.7), and the optimal primal and dual solutions for the corresponding static allocation problem are displayed in (2.8) and (2.14), respectively. Let the vector c of holding cost rates for the four job classes be

$$c = \begin{pmatrix} 1 & \frac{3}{5} & 2 & 1 \end{pmatrix}. \tag{6.3}$$

From these we compute the following values for the critical indices $\theta_i = c_i/y_i^*$ defined in (5.7), noting that job classes have been numbered according to this index in conformance with (5.11):

$$\theta = (5,\ 6,\ 8,\ 10). \tag{6.4}$$

For each $\varepsilon > 0$ we consider a discrete-review policy whose review periods are of nominal length $\ell(\varepsilon) > 0$. Here $\ell(\cdot)$ is a function that increases without bound as its argument declines to zero, growing slowly enough to insure that

$$\varepsilon\,\ell(\varepsilon) \to 0 \quad \text{as } \varepsilon \downarrow 0, \tag{6.5}$$

but growing fast enough to provide certain large deviation bounds. At the beginning of each review period the system manager allocates to the three servers some, but not necessarily all, of the jobs that are then available in the four buffers, and also specifies which activities are to be used in processing those jobs. The review period ends when all assigned jobs have been processed by the designated servers or after $\ell(\varepsilon)$ time units, whichever comes later, and let us suppose that servers simply become idle after they have completed their assignments. (This simplifies analysis, and alternative schemes that come immediately to mind are asymptotically equivalent to this one as $\varepsilon \downarrow 0$ and review periods become long.) Assignments are made in such a way that no server has an expected total processing time greater than $\ell(\varepsilon)$, and subject to this constraint, the system manager makes assignments so as to minimize the holding cost rate associated with unallocated jobs; when it is possible to allocate *all* jobs on hand subject to the stated constraint on expected total processing times, the system manager does so, striving to minimize total idleness of servers 2 and 3 as a secondary objective.

A more general form of policy, described in [7], further allows servers to work on jobs during the period in which they arrive, but restricts such activity by means of a vector $\varphi = (\varphi_1, \ldots, \varphi_m)$ of *threshold parameters*. (In [7], this parameter is called θ, but here it is relabeled to avoid confusion with the vector θ of (5.7) and (6.4).) In the more general family of policies, server time allocations to different activities during any given period must be set so that expected ending inventory in each buffer $i = 1, \ldots, m$ is at least φ_i. In our case, only jobs present in the system at the beginning of a review period are allocated to servers for processing during that period, and so in effect we have set threshold levels to one period's worth of expected arrivals, or $\varphi(\varepsilon) = \ell(\varepsilon)\lambda$, where the dependence of the threshold levels on ε has been explicitly indicated.

Using assumption 1, the complete pooling condition, and elementary large deviations theory, it can be shown that the following hold as $\varepsilon \downarrow 0$, at least under sufficiently strong distributional assumptions: the system manager very seldom chooses to use any nonbasic activities, and very seldom leaves unallocated any of the jobs on hand in buffers 2 through 4; moreover, the idleness experienced by servers 2 and 3 is negligible under heavy traffic scaling, and virtually all of the idleness experienced by server 1 occurs in periods when all jobs can be allocated. In each of these statements, phrases like "very seldom" and "virtually all" mean that the quantity in question approaches zero or one, as the case may be, under heavy traffic scaling as $\varepsilon \downarrow 0$.

Thus one sees that the defining properties of the pathwise Brownian solution (Z^*, U^*) derived in section 5 are achieved "in the limit" by the sequence of discrete-review policies described immediately above. It is then at least plausible that the scaled cost rate processes generated by that sequence of policies would satisfy (6.2). Of course, the approximating Brownian system model only becomes useful when one is able to interpret its solution in terms of limits like these, so developing a rigorous heavy traffic limit theory is a research task of high priority.

To explain in concrete terms the significance of the hypothesized result about asymptotic optimality of discrete-review policies, consider a single parallel-server system satisfying assumption 1. (This means, in particular, that the system has traffic intensity parameter $\rho^* = 1$, but the analysis extends to systems with ρ^* near 1, as explained in section 8.) Further suppose that the system manager simply wants to minimize expected total cost incurred over a finite but long time interval. Denoting by ε^{-2} the length of that interval (thus, ε is small), one can define a scaled cost rate process ζ for each scheduling policy T as in section 3, and then state the objective as follows: choose a policy T so as to

$$\text{minimize} \quad E\left[\int_0^1 \zeta(t)\,\mathrm{d}t\right]. \tag{6.6}$$

Our theory suggests that for small values of ε (that is, for long time horizons) there exists a nearly optimal policy within the class of discrete-review policies. The only free parameter of a discrete-review policy as we have described it here is the length ℓ of its planning periods, and in a practical application one would undertake a simulation study to estimate the value of ℓ which minimizes the performance measure (6.6). Of course, one can imagine many heuristic variations of the basic discrete-review strategy that might plausibly give incremental performance improvements. For example, one of the threshold parameters φ_i described above might be taken strictly smaller than $\ell\lambda_i$, allowing some class i jobs to be processed during the period of their arrival if there is adequate server capacity available. Here again, simulation studies provide the only obvious means for evaluating and choosing among the alternatives that creative thinking may suggest. The important contribution of the theory is to generate a baseline strategy that is known to be good in an asymptotic sense, from which variants can be developed and against which alternatives can be compared. This gives structure to simulation studies that might otherwise resemble groping about in complete darkness.

7. Super-server performance bounds

Consider again the problem of proving (6.2), which shows that the cost rate process ζ^* associated with our pathwise Brownian solution (Z^*, U^*) is an asymptotic bound on achievable performance in the original parallel-server system. In an earlier paper [7], which analyzed one rather special example of a parallel-server system, that result was proved in a simple way: it was first shown that achievable performance in the original model is bounded by achievable performance in a corresponding single-server model, and then that ζ^* is an asymptotic bound on the latter. In this section we first describe the naive extension of that approach to general parallel-server systems, and then explain why it cannot succeed in general, believing that this discussion will help to develop readers' intuition in several regards.

In section 3 we did not write out the constraints that must be satisfied by a vector process T of cumulative time allocations if it is to represent an admissible strategy. One such constraint is

$$A\big[T(t) - T(s)\big] \leqslant (t-s)e \quad \text{for } 0 \leqslant s \leqslant t < \infty, \tag{7.1}$$

where A is the $r \times n$ resource consumption matrix specified in section 2 and e is an r-vector of ones. This says that each server $i = 1, \ldots, m$ may allocate no more than $t - s$ time units to processing activities during the interval [s, t]. The m-dimensional

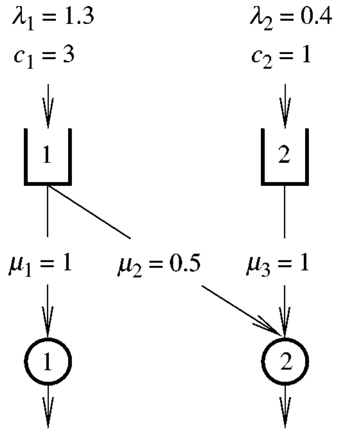

Figure 2. A two-server system with Poisson arrivals and deterministic service times.

jobcount process Q under policy T was described by (3.3), which we rewrite here for easy reference:

$$Q(t) = F^0(t) - \sum_{j=1}^{n} F^j\bigl(T_j(t)\bigr), \quad t \geqslant 0, \tag{7.2}$$

where F^0 is a vector process of cumulative arrivals into the various buffers and $F^1, \ldots, F^n$ specify cumulative outputs from the various activities given their cumulative time allocations (see section 3).

Figure 2 pictures the specific parallel-server system that was analyzed in [7]: this example is distinguished not only by its special structure but also by the special distributional assumptions described in the caption. With the given data, this model satisfies not only assumption 1 but also our complete pooling condition: that is, the two servers communicate, because they both serve class 1 in the nominal processing plan. Using the data in figure 2, readers can verify that the unique primal and dual solutions of our static allocation problem (see section 2) are given by

$$x^* = \bigl(1,\ \tfrac{3}{5},\ \tfrac{2}{5}\bigr), \quad y^* = \bigl(\tfrac{2}{3}\ \tfrac{1}{3}\bigr) \quad \text{and} \quad z^* = \bigl(\tfrac{2}{3}\ \tfrac{1}{3}\bigr). \tag{7.3}$$

The pathwise solution of the limiting Brownian control problem was derived in [7], and it was shown that the ideal system behavior described by that solution can be approached in the heavy traffic limit by using discrete-review policies of the kind described in the previous section.

In [7] the pathwise optimal cost rate process ζ^* arising in the Brownian system model was interpreted in terms of a single pooled resource, or super-server, that combines the capabilities of the two servers in figure 2. Naively translated into the general parallel-server context considered in this paper, and using the notation developed here, the bounding super-server model is described as follows. First, imagining a single super-server that is able to undertake all n of the original processing activities, and denoting by $\widehat{T} = \{\widehat{T}(t),\ t \geqslant 0\}$ the n-dimensional, nondecreasing process of cumulative time allocations for the super-server, one replaces (7.1) by

$$e \cdot \big[\widehat{T}(t) - \widehat{T}(s)\big] \leqslant t - s \quad \text{for } 0 \leqslant s \leqslant t < \infty. \tag{7.4}$$

(As in section 3, an admissible policy $\widehat{T}$ for the super-server must be non-anticipating in a certain sense and must satisfy other restrictions as well, but those aspects of the model formulation need not be spelled out here.) Second, for each activity j we define an "acceleration factor"

$$a_j = \frac{1}{z^*_{k(j)}}, \quad j = 1, \ldots, n. \tag{7.5}$$

Finally, (7.2) is replaced by the following definition of the jobcount process $\widehat{Q}$ under super-server policy $\widehat{T}$:

$$\widehat{Q}(t) = F^0(t) - \sum_{j=1}^{n} F^j\big(a_j \widehat{T}_j(t)\big), \quad t \geqslant 0. \tag{7.6}$$

This means that one unit of super-server effort devoted to activity j produces exactly the same effect as a_j units of effort devoted to that activity by server $k(j)$ in our original model. To see that optimal performance in the super-server model really does bound that in our original model, one need only note the following: given any policy T satisfying (7.1) in the original model, we can define a corresponding super-server policy $\widehat{T}$ via

$$\widehat{T}_j(t) = z^*_{k(j)} T_j(t) \quad \text{for } t \geqslant 0 \text{ and } j = 1, \ldots, n. \tag{7.7}$$

Because $z^*_1 + \cdots + z^*_r = 1$, it follows from (7.1), (7.7) and the definition of A that (7.4) holds, and the jobcount process $\widehat{Q}$ derived from $\widehat{T}$ via (7.6) is identical to the jobcount process Q derived from T via (7.2).

Having established that the super-server model bounds our original model, one now asks whether the two are "asymptotically equivalent" under heavy traffic scaling. For the two-server example described in figure 2, that equivalence was rigorously proved in [7], but it depends on the assumption that all service times are deterministic. (A more general but still restrictive condition which is believed to assure the asymptotic equivalence of the two models will be stated shortly.) To see that the equivalence is not general, consider the two-server model described in figure 2, modified (only) in the following way: when server 1 processes jobs from buffer 1 (this is activity 1), the service times are exponentially distributed rather than deterministic, but with unit mean as before. From (7.3) and (7.5), the acceleration factors for the three activities in the super-server model are found to be $a_1 = \frac{3}{2}$, $a_2 = 3$ and $a_3 = 3$, respectively. Looking only at activities 1 and 2, we then have the following: the super-server may either process jobs from buffer 1 in deterministic fashion at rate $a_2\mu_2 = \frac{3}{2}$ (this is activity 2), or process them with exponential service times at average rate $a_1\mu_1 = \frac{3}{2}$ (this is activity 1). The pathwise bound ζ^* that we derived in section 3 represents a heavy traffic limit under policies which mix those two modes of processing in proportions fixed by the nominal processing plan x^*, but here the super-server model has been defined in such a way that activity 2 (the deterministic mode) can be used exclusively for the processing of class 1, which reduces variability. Given that capability, and all the other special features of the two-server example, one can show the following: by using only activity 2 to process class 1, and always giving priority to class 1 over class 2, the super-server can achieve a limiting scaled cost rate process ζ^{o} that is a driftless one-dimensional reflected Brownian motion like ζ^*, but which has strictly smaller variance parameter than does ζ^*. The lower-variance process ζ^{o} is strictly stochastically smaller than ζ^* at every time $t > 0$.

In light of this example it is natural to consider parallel-server systems that have the following special structure. Suppose that the service time distributions for activities which process any given job class are identical except for activity-specific scale factors. It might be, for example, that service time distributions for activities associated with class 1 are all deterministic, regardless of which server undertakes the activity, that service time distributions associated with class 2 are all exponential, and so forth, but the mean service time may differ according to which server does the processing. If a parallel-server system has this special structure, we believe that the Brownian cost rate process ζ^* does provide an asymptotic bound on achievable performance in its naive super-server analog, and thus bounding the original system by that super-server model is useful for purposes of heavy traffic theory. Proving this proposition does not appear to be very difficult, in fact, but the task will not be attempted here.

Having argued earlier that ζ^* does not generally provide an asymptotic bound on achievable performance in the naive super-server model, we now describe a restricted version of the super-server model for which that asymptotic bound *is* generally valid. For each $i = 1, \ldots, m$ let $B(i)$ be the set of all basic activities $j \in \{1, \ldots, b\}$ such that $i(j) = i$. That is, $B(i)$ consists of those basic activities which process jobs of class i. Now let

$$\beta_i = \sum_{j \in B(i)} z^*_{k(j)} x^*_j \quad \text{for } i = 1, \ldots, m, \tag{7.8}$$

and

$$\gamma_j = \frac{z^*_{k(j)} x^*_j}{\beta_i} \quad \text{for } j \in B(i) \ \text{ and } \ i = 1, \ldots, m. \tag{7.9}$$

Thus the constants γ_j are strictly positive and they sum to one over each of the sets $B(i)$. Let the super-server model be as described above except that now its time allocations are restricted by the requirement that (for all $t \geqslant 0$)

$$\widehat{T}_j(t) = \gamma_j \tau_i(t) \quad \text{for } j \in B(i) \text{ and } i = 1, \ldots, m, \tag{7.10}$$

where $\tau_i = \{\tau_i(t),\ t \geqslant 0\}$ is some nondecreasing process. One interprets $\tau_i(t)$ as the cumulative time allocated by the super-server to processing class i jobs over the interval [0, t], and (7.10) requires that each increment of that cumulative allocation be divided among the basic activities $j \in B(i)$ in the fixed proportions $\{\gamma_j,\ j \in B(i)\}$. Hereafter the m-dimensional vector process τ with components $\tau_1, \ldots, \tau_m$ will be referred to as a *class-level time allocation*, or *class-level strategy*, for the super-server. Of course, (7.4) requires that τ satisfy

$$e \cdot \big[\tau(t) - \tau(s)\big] \leqslant t - s \quad \text{for } 0 \leqslant s \leqslant t < \infty, \tag{7.11}$$

where e is an m-vector of ones. (Again we observe that an admissible class-level strategy τ must satisfy other constraints as well, but those need not be spelled out for purposes of this discussion.) Substituting (7.10) into (7.6), one has the following representation for the super-server jobcount process $\widehat{Q}$ under class-level strategy τ:

$$\widehat{Q}(t) = F^0(t) - \sum_{i=1}^{m} \widehat{F}^i\big(\tau_i(t)\big), \quad t \geqslant 0, \tag{7.12}$$

where

$$\widehat{F}^i(t) = \sum_{j \in B(i)} F^j(a_j \gamma_j t) \quad \text{for } i = 1, \ldots, m. \tag{7.13}$$

Condition (7.10) says that the super-server is obliged to process class i jobs by means of a *composite basic activity* which actually distributes the class-level time allocation in fixed proportions among basic activities in $B(i)$, and $\widehat{F}^i = \{\widehat{F}^i(t),\ t \geqslant 0\}$ is the cumulative output process achieved when the super-server devotes all of its time to that composite activity. Recalling from (7.5) that $a_j = 1/z^*_{k(j)}$, we have from (7.9) and (7.13) that

$$\widehat{F}^i(t) = \sum_{j \in B(i)} F^j\big(\beta_i^{-1} x_j^* t\big) \quad \text{for } i = 1, \ldots, m. \tag{7.14}$$

Denoting by $\hat{\mu}_i$ the average service rate when all super-server time is devoted to composite basic activity i, we have from (7.14) that

$$\hat{\mu}_i = \sum_{j \in B(i)} \beta_i^{-1} x_j^* \mu_j \quad \text{for } i = 1, \ldots, m. \tag{7.15}$$

Of course, our primal constraints (2.2) require that the nominal processing plan x^* satisfy

$$\sum_{j \in B(i)} \mu_j x_j^* = \lambda_i \quad \text{for } i = 1, \ldots, m, \tag{7.16}$$

and (7.15) and (7.16) together give

$$\hat{\mu}_i = \lambda_i \beta_i^{-1} \quad \text{for } i = 1, \ldots, m. \tag{7.17}$$

Finally, recall from (2.15) that our optimal dual variables (y^*, z^*) satisfy the complementary slackness condition $\mu_j y^*_{i(j)} = z^*_{k(j)}$ for $j = 1, \ldots, b$. Substituting that relationship into (7.8) and again using (7.16), one has $\beta_i = y_i^* \lambda_i$, and thus, (7.17) reduces to

$$\hat{\mu}_i = \frac{1}{y_i^*} \quad \text{for } i = 1, \ldots, m. \tag{7.18}$$

In the restricted super-server model described immediately above, it is relatively easy to prove the following: if the classes are ranked in increasing order of the index $c_i \hat{\mu}_i = c_i / y_i^*$, and if higher ranked classes are given priority over lower ranked ones, then the associated cost rate process $C(t) = c \cdot \widehat{Q}(t)$ converges weakly after heavy traffic scaling to our pathwise Brownian solution ζ^* (see section 5); moreover, if ζ is the weak limit of the scaled cost rate process under any other policy or family of policies, then one has $\zeta(t) \geqslant \zeta^*(t)$ for all $t \geqslant 0$ as in (5.10). Unfortunately, it is by no means obvious that optimal performance in the restricted super-server model dominates optimal performance in our original parallel-server model, even asymptotically under heavy traffic scaling. That is, when the super-server is required to process each job class by means of the composite basic activity described above, it is no longer obvious that its best achievable performance dominates the best achievable performance in our original model, although we believe that to be true in the heavy traffic limit.

8. Concluding remarks

Let us consider now the problem of approximating a parallel-server system whose traffic intensity parameter (see section 2) is near 1 but not exactly equal to 1. In this discussion we shall first speak in terms of approximating a single given system, specializing appropriately the more general exposition in [8, section 5], and then describe the relationship of that approximation to heavy traffic limit theory. To conform with the notational conventions employed in [8], we shall re-use the letter θ with a new meaning in this section.

Consider a parallel-server system with first-order data (R, A, λ), also taking as given a small parameter $\varepsilon > 0$ whose magnitude reflects the time scale that is relevant for purposes of performance measurement (see section 3). Let us suppose that there exists a strictly positive m-vector λ^* such that assumption 1 holds with λ^* in place of λ, and which is close to λ in the following sense:

$$\text{all components of the } m\text{-vector } \theta = \varepsilon^{-1}\big(\lambda - \lambda^*\big) \text{ are moderate in value.} \tag{8.1}$$

This means, among other things, that the static allocation problem with data (R, A, λ^*) has a unique solution (x^*, ρ^*) and that $\rho^* = 1$. Again we call x^* the nominal processing plan, observing that it fully allocates all server capacity (that is, $Ax^* = e$) and that its associated vector of average output rates from the various buffers is $Rx^* = \lambda^*$ rather than $Rx^* = \lambda$. Deviation controls are defined in terms of x^* via (3.5) under any given policy, and scaled processes are defined using the given parameter ε via (3.8)–(3.11). Proceeding exactly as in [6], this leads to an approximating Brownian system model which is identical to the one specified by (4.2)–(4.6) except that X is now an m-dimensional Brownian motion with covariance matrix Γ and with drift vector θ defined by (8.1). Readers should note that the vector x^*, which is derived from λ^* rather than λ, plays a central role in calculating data of the Brownian system model, and that λ comes into play only in computing the drift vector θ via (8.1). Propositions 4 and 5, which give the condition for complete pooling in a Brownian system model and identify the resulting pathwise solution, respectively, are applied using (R, A, λ^*) rather than (R, A, λ), because the relevant data of the Brownian model have been computed from λ^* rather than λ. (Note that the drift vector of the Brownian motion X plays no role in either proposition.)

One scenario satisfying the assumptions listed in the previous paragraph, and probably the only scenario that one need consider from a practical standpoint, is the following. Suppose that our first-order data (R, A, λ) are such that parts (i) and (iii) of assumption 1 hold, but the traffic intensity parameter ρ^* derived from the static allocation problem is not exactly 1. Then we can simply take $\lambda^* = \lambda/\rho^*$, and (8.1) amounts to the requirement that $\varepsilon^{-1}(1-\rho^*)$ be of moderate absolute magnitude, which is a familiar condition from classic heavy traffic theory. In this case, of course, the drift vector θ in (8.1) is just λ multiplied by the real constant $\kappa = (\rho^*-1)/\varepsilon\rho^*$, and it is clear why $|\kappa|$ must be moderate if we are to obtain a meaningful Brownian approximation from the scaling constant ε that is here taken as given: if $|\kappa|$ is not moderate, then all components of the drift vector θ will be either very large (positive) or else very negative, and that extreme drift will overwhelm all other problem elements.

To develop a rigorous heavy traffic limit theory that justifies the approximating Brownian system model described above, one may hypothesize a family of parallel-server systems parameterized by $\varepsilon > 0$, in which all problem data other than the vector λ of average arrival rates are fixed. Translating (8.1) into a formal notion of a heavy traffic limit, we assume that $\lambda^\varepsilon \to \lambda^*$ as $\varepsilon \downarrow 0$, where λ^ε has the obvious meaning, λ^* is a strictly positive vector such that 1 is satisfied with λ^* in place of λ, and moreover,

$$\varepsilon^{-1}\bigl(\lambda^\varepsilon - \lambda^*\bigr) \to \theta \quad \text{as } \varepsilon \downarrow 0, \tag{8.2}$$

where θ is an m-vector whose components are moderate in absolute magnitude. The “limiting Brownian system model” for this family of parallel-server systems is as described above: the underlying Brownian motion X has drift vector θ and all of its other data are computed using the vector λ^* of limiting average arrival rates. To prove the asymptotic optimality of a family of policies parameterized by $\varepsilon > 0$, one may then seek to prove (6.1) and (6.2), exactly as in the case of a single system with $\rho^* = 1$ but with scaling parameter $\varepsilon \downarrow 0$. It is this framework that will be used in our future investigation of discrete-review strategies.

Earlier it was said that part of proposition 4 and all of proposition 5 are special cases of results proved by Laws [16]. To be more specific, Laws considered dynamic control policies, including both routing and sequencing decisions, for a class of queueing network models that are special in some regards but still much more general than parallel-server systems. (Viewed in the context of stochastic network theory, the distinguishing feature of a parallel-server system is that the route of each arriving job consists of just one operation.) He defined heavy traffic for such a network by means of a linear program that is equivalent to our static allocation problem, and focused on heavy traffic limits such that a dual linear program equivalent to (2.9)–(2.12) has a unique solution: in his language this is the case of a heavy traffic limit with "a single binding generalized cut constraint". Writing out the natural Brownian approximation for his dynamic control problem with only heuristic justification, just as we have done here, he then showed that the Brownian control problem has a one-dimensional equivalent workload formulation, and he derived the pathwise solution of that one-dimensional problem. Expressing Laws' results in the language of this paper, he proved that condition (i) of propositions 3 and 4 (uniqueness of the optimal dual solution) implies $d = 1$ in a much more general setting, and he also proved a more general version of our proposition 5. Both of those results have been further generalized in [6,8]. For the special case of parallel-server systems, we have shown in proposition 4 that the limiting Brownian control problem is essentially one-dimensional if and only if all servers communicate in the sense of section 2, which is a simple and intuitive condition for "complete pooling" that seems to have no analog in a general network setting. Another benefit of our specialization to parallel-server systems, which will only be demonstrated in the future study of discrete-review policies that was previewed in section 6, is that one can identify without great difficulty a concrete means of approaching, in the heavy traffic limit, the ideal system behavior described by the pathwise Brownian solution. In contrast, Laws [16] was unable to propose a means of achieving his pathwise solutions in the more general network context, and that unsolved problem was highlighted in a subsequent survey paper [12].

A recent paper by Kushner and Chen [13] is devoted to heavy traffic analysis of parallel-server systems in which each server has a "dedicated" input stream that demands almost all of its capacity. What gives their problem its flavor is that there also exist "assignable" input streams which have much lower arrival rates and may be processed by any of several servers. In the heavy traffic limit where each server is fully loaded by its own dedicated stream, the assignable streams represent a vanishingly small fraction of total input, and yet the way in which they are distributed among the servers (dynamically, based on observed system status) is still significant from the standpoint of system performance. The Kushner–Chen heavy traffic regime is in a sense the diametric opposite of the one on which we focus in this paper, because the domination of each server by a dedicated input stream eliminates any possibility of resource pooling in the heavy traffic limit. Kushner and Chen obtain a limiting Brownian control problem that is unlike those encountered in any prior work on heavy traffic control problems but which is still amenable to numerical solution by the Markov chain approximation method [14]. In their setting it is apparently a straightforward matter to translate optimal Brownian controls (computed numerically) into implementable strategies for the original queueing system, and the authors are able to rigorously prove asymptotic optimality of those strategies.

References

[1] D. Bertsimas and J.N. Tsitsiklis, *Introduction to Linear Optimization* (Athena Scientific, Belmont, MA, 1997).

[2] P.B. Chevalier and L.M. Wein, Scheduling networks of queues: Heavy traffic analysis of a multistation closed network, Oper. Res. 41 (1993) 743–758.

[3] D.R. Cox and W.L. Smith, *Queues* (Methuen, London, 1961).

[4] J.M. Harrison, *Brownian Motion and Stochastic Flow Systems* (Wiley, New York, 1985).

[5] J.M. Harrison, Brownian models of queueing networks with heterogeneous customer populations, in: *Stochastic Differential Systems, Stochastic Control Theory and Applications*, eds. W. Fleming and P.L. Lions (Springer, New York, 1988) pp. 147–186.

[6] J.M. Harrison, The BIGSTEP approach to flow management in stochastic processing networks, in: *Stochastic Networks: Theory and Applications*, eds. F. Kelly, S. Zachary and I. Ziendins (Oxford Univ. Press, Oxford, 1996).

[7] J.M. Harrison, Heavy traffic analysis of a system with parallel servers: Asymptotic optimality of discrete-review policies, Ann. Appl. Probab. 8 (1998) 822–848.

[8] J.M. Harrison, Brownian models of open processing networks: Canonical representation of workload (1997) submitted for publication.

[9] J.M. Harrison and J.A. Van Mieghem, Dynamic control of Brownian networks: State space collapse and equivalent workload formulations, Ann. Appl. Probab. 7 (1997) 747–771.

[10] J.M. Harrison and L.M. Wein, Scheduling networks of queues: Heavy traffic analysis of a simple open network, Queueing Systems 5 (1989) 265–280.

[11] J.M. Harrison and L.M. Wein, Scheduling networks of queues: Heavy traffic analysis of a two-station closed network, Oper. Res. 38 (1990) 1052–1064.

[12] F.P. Kelly and C.N. Laws, Dynamic routing in open queueing networks: Brownian models, cut constraints and resource pooling, Queueing Systems 13 (1993) 47–86.

[13] H.J. Kushner and Y.N. Chen, Optimal control of assignment of jobs to processors under heavy traffic, preprint, Department of Applied Mathematics, Brown University (1998).

[14] H.J. Kushner and P. Dupuis, *Numerical Methods for Stochastic Control Problems in Continuous Time* (Springer, Berlin/New York, 1992).

[15] H.J. Kushner and L.F. Martins, Routing and singular control for queueing networks in heavy traffic, SIAM J. Control Optim. 28 (1990) 1209–1233.

[16] C.N. Laws, Resource pooling in queueing networks with dynamic routing, Adv. in Appl. Probab. 24 (1992) 699–726.

[17] C.N. Laws and G.M. Louth, Dynamic scheduling of a four-station queueing network, Probab. Engrg. Inform. Sci. 4 (1990) 131–156.

[18] L.M. Wein, Scheduling networks of queues: Heavy traffic analysis of a two-station network with controllable inputs, Oper. Res. 38 (1990) 1065–1078.

[19] L.M. Wein, Brownian networks with discretionary routing, Oper. Res. 39 (1991) 322–340.

[20] L.M. Wein, Scheduling networks of queues: Heavy traffic analysis of a multistation network with controllable inputs, Oper. Res. 40 (1992) S312–S334.