Optimal Stochastic Scheduling in Multiclass Parallel Queues

Jay Sethuraman
Operations Research Center
Massachusetts Institute of Technology
Cambridge, MA 02139
jayc@mit.edu

Mark S. Squillante
IBM Research Division
Thomas J. Watson Research Center
Yorktown Heights, NY 10598
mss@watson.ibm.com

Abstract

In this paper we consider the problem of scheduling different classes of customers on multiple distributed servers to minimize an objective function based on per-class mean response times. This problem arises in a wide range of distributed systems, networks and applications. Within the context of our model, we observe that the optimal sequencing strategy at each of the servers is a simple static priority policy. Using this observation, we argue that the globally optimal scheduling problem reduces to finding an optimal routing matrix under this sequencing policy. We formulate the latter problem as a nonlinear programming problem and show that any interior local minimum is a global minimum, which significantly simplifies the solution of the optimization problem. In the case of Poisson arrivals, we provide an optimal scheduling strategy that also tends to minimize a function of the per-class response time variances. Applying our analysis to various static instances of the general problem leads us to rederive many results, yielding simple approximation algorithms whose guarantees match the best known results.

1 Introduction

The fundamental problem of scheduling a set of distributed resources among different classes of customers to achieve some performance objective has received and continues to receive considerable attention in the research literature. This is motivated by problems arising in a wide range of distributed computer applications and system environments, as well as communication network environments. A particular recent instance of the general problem is motivated by scalable Web server systems where incoming Web requests are immediately routed to one of a set of computer nodes by a high-speed router, and each node independently executes the customers assigned to it following a local sequencing algorithm [6, 9].

We consider the problem of scheduling different classes of customers on multiple distributed heterogeneous servers to minimize an objective function based on per-class mean response times. This optimal scheduling problem consists of two distinct decisions: (*i*) the allocation of customers to the parallel servers; and (*ii*) the order of execution for the customers at each server. The first decision has the flavor of a global *load-balancing* optimization problem in which the customers are distributed among the multiple heterogeneous servers to minimize the response-time objective function. The second decision is local in nature and consists of solving an optimal *sequencing* problem: given a mix of customers at a server, determine the best order of service for the queued customers to satisfy the global objective. In our present study we consider the structure of the optimal solution to the general problem of interest under the following restrictions on these two decisions:

- **Allocation**: Customers are allocated to servers in a *probabilistic* manner; i.e., immediately upon arrival, a customer is assigned to a server based on a matrix of routing probabilities. This is often called *random splitting* [26].
- **Sequencing**: The sequencing strategy is non-anticipative (i.e., does not require knowledge of the future), work-conserving (i.e., does not idle when there is work to do) and non-preemptive (i.e., the execution of a customer cannot be interrupted and subsequently resumed).

The objective considered in this paper is to globally minimize a linear function of the per-class mean response times. Similar techniques can be used to minimize a linear function of the per-class mean waiting times. Throughout this paper we use the terms *customer* and *server* in order to be completely general and not restricted to any particular application area.

Our analysis of this optimal scheduling problem begins with the observation that the optimal sequencing strategy at each of the servers is a simple static priority policy. Using this observation, we argue that the globally optimal scheduling problem reduces to finding an optimal routing matrix under this sequencing policy. We formulate the latter problem as a nonlinear programming problem and show that it has at most one solution in the interior of the feasible domain and that any local minimum in the interior is a global minimum. This result significantly simplifies the solution of the general optimal scheduling problem. We first restrict our attention to Poisson arrivals, in which case we derive an optimal scheduling policy that also tends to minimize a function of the per-class response time variances. We then consider the case of general arrivals by developing a fluid-model formulation of the optimization problem and deriving an analogous set of results. The use of fluid models as approximations for queueing systems, often within the context of optimal control, has received and continues to receive considerable attention in the research literature; e.g., see [11, 4, 1, 13] and the references cited therein.

Related scheduling problems have been examined in the research literature. Our scheduling problem is consistent with or a generalization of the problems considered in [2, 16, 3, 6, 9, 5] and the relevant references therein. A number of these studies [6, 9, 5] have analyzed the performance of specific policies, as opposed to obtaining the globally optimal solution. Borst [3] considers the globally

SIGMETRICS '99 5/99 Atlanta, Georgia, USA
© 1999 ACM 1-58113-083-X/99/0004...$5.00

suboptimal scheduling problem of finding the optimal routing matrix under an FCFS sequencing policy at each server, within the context of a system model similar to the Poisson arrival instance of the model assumed in our study. Our analysis addresses the globally optimal scheduling problem using different methods than those of Borst and yielding a scheduling strategy that also tends to have better per-class response time variance properties. Furthermore, our analysis of the fluid version of the optimization problem establishes a corresponding set of results that are not restricted to Poisson arrivals. The allocation component of the optimal scheduling problem is somewhat related to a global *load-balancing* optimization problem that has received considerable attention in the literature; e.g., see [25, 26, 2, 16] and the references cited therein. Ross and Yao [16] consider a problem that is similar to a single-class instance of the problem studied in this paper, with the addition of a dedicated independent stream of customer arrivals to each server having non-preemptive priority over the other customers. Bonomi and Kumar [2] consider a model similar to that in [16] but with additional restrictions, and in both studies the objective is to minimize the average response time taken over the two sets of customers where each arrival stream is a Poisson process. We note that additional linear equalities, which includes the dedicated stream of arrivals in [2, 16], can be easily accommodated in our approach. Hence, the scheduling problem, the models and the class of objective functions considered in our study are more general than those examined in [2, 16]. Moreover, we use different methods than those proposed in [2, 16] to solve the general optimal scheduling problem, and in the case of Poisson arrivals we further address some properties of per-class response time variance.

We also consider various static instances of the general optimal scheduling problem where a finite set of customers arrive at time 0 and there are no other arrivals, in which cases the stochastic problems reduce to the corresponding deterministic scheduling problems (where the processing times are replaced by their expected values) without loss of generality. Following our solution approach for these special cases leads us to rederive many results in a fairly elegant manner, yielding simple approximation algorithms whose guarantees match the best known results. Our approximation algorithms are based on the use of randomized rounding on a convex relaxation, which is the first use of such a relaxation in the scheduling literature to our knowledge. We obtain an ϵ-improvement over the previously known algorithm due to Schulz and Skutella [17]. For the special case in which all the servers are identical, our analysis provides an optimal closed-form solution to a simpler convex relaxation. A derandomized version of our algorithm for this case also yields the algorithm due to Kawaguchi and Kyan [10]. Furthermore, we believe that improvements in the approximation guarantees for some of the special cases considered will be possible by exploiting the convex programming techniques of our approach.

The allocation strategy considered in this paper is *static* in the sense that the routing probabilities do not change dynamically with time nor do they depend upon the server queue lengths. While dynamic allocation policies have the potential to outperform static policies [12, 7, 23], implementing a dynamic policy can be nontrivial and these policies can incur considerable overheads. Static policies may therefore be preferable in certain practical situations, such as the distributed environments motivating our present study [6, 9, 5]. The use of our optimal scheduling solution in practice can also consist of periodic adjustments of the routing matrix of the allocation strategy with changes in the system environment, such as variations in the workload. Moreover, given an optimal routing matrix, one can use an equivalent deterministic version of the probabilistic routing scheme to obtain lower (response time) variance properties in a real system. This approach is consistent with that taken in [6, 9] where a deterministic implementation of a static load-balancing policy is used together with each computer node periodically informing the router of changes in its load.

The sequencing strategy considered in this paper is restricted to non-preemptive policies. While preemption has the potential to improve mean response times, it can involve considerable overhead in practice. We note, however, that the results presented in this paper directly hold for preemptive sequencing strategies under exponential service time distributions. Furthermore, it can be established that the optimal preemptive sequencing policy is a dynamic indexing scheme based on the remaining service times for the customers [19]. The rest of our results should then hold together with this sequencing strategy, which is the subject of future work.

The remainder of this paper is organized as follows. We first consider the general stochastic scheduling problem under independent Poisson arrival streams. Then in Section 3 we remove this assumption of Poisson arrivals and consider a fluid-model formulation of the general stochastic scheduling problem. Section 4 presents an analysis of static instances of the general problem, and our concluding remarks are provided in Section 5.

2 Poisson Arrival Case

In this section we define more precisely the optimal scheduling problem of interest under the assumption of Poisson arrivals, for which we derive an efficient solution. We first present the corresponding system model and define the linear mean response time objective function considered in our study. An analysis of the sequencing and random splitting aspects of the optimal scheduling problem is then derived in Sections 2.2 and 2.3, respectively. We end this section by developing an equivalent optimal scheduling policy that tends to also minimize a function of the per-class response time variances.

2.1 The Model

We consider a system model consisting of K independent customer classes and N heterogeneous parallel servers. Throughout this paper, we will use i to index the customer classes and j to index the servers under the constraints $i = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, N$, unless noted otherwise. Customers of class i arrive to the system from a Poisson source with rate λ_i. The total customer arrival rate is given by $\lambda = \sum_{i=1}^{K} \lambda_i$. Each customer is routed to one of the servers immediately upon its arrival according to a probability matrix $\mathbf{P} \equiv [p_{ij}]_{1 \le i \le K; 1 \le j \le N}$, independent of all else; i.e., a class i customer arrival is independently routed to server j with probability p_{ij}. The rate of customer arrivals to server j is therefore given by $\sum_{i=1}^{K} \lambda_i p_{ij}$. The service time of a class i customer when executed on server j has a general distribution on $\mathbb{R}^+$ given by $F_{ij}(\cdot)$, mutually independent of the arrival and routing processes. We assume that the servers differ only in their speeds, and we use s_j to denote the speed of server j. Hence, $F_{ij}(t) = F_i(s_j t)$; i.e., the base service-time distribution of class i customers is $F_i(t)$ with server j having service rate s_j. Once started, each customer is executed to completion without interruption; i.e., there is no preemption.

Most of our analysis requires only the first and second moments of the base service-time distributions $F_i(\cdot)$, which we respectively denote by x_i and $x_i^{(2)}$. When a class i customer is executed on server j, the expected service time is given by $x_{ij} = x_i / s_j$ and the second moment of the service time is given by $x_{ij}^{(2)} = x_i^{(2)} / s_j^2$. Let $\rho_i = \lambda_i x_i$ be the traffic intensity for class i. The traffic intensity at server j can then be expressed as $\phi_j = \sum_{i=1}^{K} \rho_i p_{ij}$. Since s_j is the capacity of server j, necessary and sufficient conditions for stability (i.e., finite expected response times) are given by

$$\phi_j = \sum_{i=1}^{K} \rho_i p_{ij} < s_j, \qquad j = 1, 2, \ldots, N. \qquad (1)$$

The total traffic intensity of the system is $\rho = \sum_{j=1}^{N} \phi_j$, and the total capacity of the system is $S = \sum_{j=1}^{N} s_j$. It then follows directly from (1) that the system is stable as long as $\rho < S$, which we assume to be the case in what follows.

Objective Function. A positive weight c_i is associated with customers of class i. The scheduling objective of interest in this paper is to minimize the linear function

$$\sum_{i=1}^{K} c_i \frac{\lambda_i}{\lambda} T_i \tag{2}$$

within the context of the above system model, where T_i is the expected total amount of time spent in the system by class i customers. We shall refer to T_i as the mean response time for class i.

2.2 Optimal Sequencing

Let us initially suppose that we have obtained the routing matrix $\mathbf{P}^*$ for the globally optimal scheduling policy. It follows from the properties of the class of allocation policies under consideration that the servers are equivalent to multiclass $M/G/1$ queues. In particular, server j is a K-class non-preemptive $M/G/1$ queue with per-class arrival rates $\lambda_i p_{ij}$. The mean per-class response times will depend on the sequencing strategy employed at the server.

Our interest is in finding the sequencing strategy that minimizes the objective function in equation (2). Smith [21] showed that the optimal sequencing strategy which minimizes such a linear function of expected response times in a multiclass $M/G/1$ queue is a fixed priority policy, often referred to as the *$c\mu$ rule*. Specifically, he established the following result.

Proposition 2.1 *Consider a K-class non-preemptive $M/G/1$ queue, with class i arrival rate λ_i and class i service time distribution G_i having mean x_i. Let T_i denote the expected response time of class i customers, let c_i be a positive constant associated with class i, and define $\lambda = \sum_{i=1}^{K} \lambda_i$. Then, the scheduling policy Π that gives priority to class i customers over class k whenever $c_i/x_i \geq c_k/x_k$ minimizes $\sum_{i=1}^{K} c_i \frac{\lambda_i}{\lambda} T_i$.*

Observe from Proposition 2.1 that the scheduling policy Π which minimizes the response time objective function in (2) is the same for all servers. This follows from the assumption that the servers are identical except for their speed and thus no class is given "preferential treatment" by any server. For convenience, and without loss of generality, we assume that the customer classes are labeled such that

$$\frac{c_1}{x_1} \geq \frac{c_2}{x_2} \geq \ldots \geq \frac{c_K}{x_K}.$$

This labeling ensures that under the optimal scheduling policy the priority ordering of the customer classes at any server follows the index order: class k is given priority over class ℓ if $k < \ell$. Since we assume that servers differ only in their speed, the optimal ordering $<_j$ for server j is independent of j and is denoted by $<$. Then, from standard queueing theory (e.g., refer to [11], noting that our priority ordering of the customer classes is the opposite of what is considered in [11]), the mean waiting time of class k customers in the multiclass $M/G/1$ queue of Proposition 2.1 is given by

$$W_k = \frac{\sum_{i=1}^{K} \lambda_i x_i^{(2)}}{2(1 - \sum_{\ell<k} \rho_\ell)(1 - \sum_{\ell \leq k} \rho_\ell)}, \tag{3}$$

and the corresponding mean class-k response time can be expressed as

$$T_k = W_k + x_k. \tag{4}$$

This simple analysis shows that our optimal scheduling problem is as hard as finding the optimal routing matrix $\mathbf{P}^*$. Observe also that the optimal sequencing policy is identical for all servers and does not depend on $\mathbf{P}^*$. The optimal sequencing policy instead depends only on the first moment of the service time distributions of the customers. Knowing the optimal sequencing policy enables us to express the expected response times of the customer classes in terms of $\mathbf{P}^*$. Thus, the globally optimal scheduling problem reduces to the problem of finding the optimal routing matrix under the optimal sequencing policy, and this problem can be posed as an optimization problem which is addressed in the next section.

2.3 Optimal Random Splitting

Based on the above analysis, we formulate the problem of finding an optimal routing matrix as a nonlinear programming problem. Let each server order all of the customers assigned to it according to the priority rule of Proposition 2.1 in Section 2.2.

Note that the arrival rate of class i customers to server j is $\lambda_i p_{ij}$. From equation (3), the mean waiting time of a class i customer at server j can then be expressed as

$$W_{ij} = \frac{\sum_{k=1}^{K} \lambda_k p_{kj} x_{kj}^{(2)}}{2(1 - \sum_{k:k<i} \rho_{kj})(1 - \sum_{k:k \leq i} \rho_{kj})}, \tag{5}$$

where $\rho_{kj} = \lambda_k p_{kj} x_{kj}$. We therefore have

$$\begin{aligned} W_i &= \sum_{j=1}^{N} \frac{\lambda_i p_{ij}}{\lambda_i} W_{ij} \\ &= \sum_{j=1}^{N} p_{ij} \left(\frac{\sum_{k=1}^{K} \lambda_k p_{kj} x_{kj}^{(2)}}{2(1 - \sum_{k:k<i} \rho_{kj})(1 - \sum_{k:k \leq i} \rho_{kj})} \right), \end{aligned}$$

from which together with (4) we obtain

$$\begin{aligned} T_i &= \sum_{j=1}^{N} p_{ij}(W_{ij} + x_{ij}) = W_i + \sum_{j=1}^{N} x_{ij} p_{ij} \\ &= \sum_{j=1}^{N} p_{ij} \left(\frac{\sum_{k=1}^{K} \lambda_k p_{kj} x_{kj}^{(2)}}{2(1 - \sum_{k<i} \rho_{kj})(1 - \sum_{k \leq i} \rho_{kj})} + x_{ij} \right). \end{aligned} \tag{6}$$

Using the analysis and observations of Sections 2.1 and 2.2 together with (1) and (6), our optimal scheduling problem reduces to finding the optimal routing matrix $\mathbf{P}^* \equiv [p_{ij}^*]_{1 \leq i \leq K; 1 \leq j \leq N}$ that solves the following optimization problem:

$$\begin{aligned} \text{(RS)} \quad \min \quad & \sum_{i=1}^{K} c_i \frac{\lambda_i}{\lambda} \sum_{j=1}^{N} p_{ij} \cdot \\ & \left(\frac{\sum_{k=1}^{K} \lambda_k p_{kj} x_{kj}^{(2)}}{2(1 - \sum_{k:k<i} \rho_{kj})(1 - \sum_{k:k \leq i} \rho_{kj})} + x_{ij} \right) \\ \text{s.t.} \quad & \\ & \sum_{j=1}^{N} p_{ij} = 1, \ \forall i, \\ & \sum_{i=1}^{K} \lambda_i x_i p_{ij} < s_j, \ \forall j, \\ & p_{ij} \geq 0, \ \forall i, j. \end{aligned}$$

The optimization problem (RS) is a nonlinear programming problem and it appears to be difficult to solve in general. However, we establish the following theorem which considerably simplifies the task of solving this optimization problem.

Theorem 2.2 *Any local minimum of problem* (RS) *in the interior of the feasible domain is a global minimum.*

Proof Sketch: Let $\mathcal{F}(\mathbf{p})$ denote our objective function, and thus

$$\mathcal{F}(\mathbf{p}) = \frac{1}{\lambda} \sum_{j=1}^{N} \frac{\sum_{i=1}^{K} \lambda_i p_{ij} x_{ij}}{2} \sum_{i=1}^{K} \frac{c_i \lambda_i p_{ij}}{(1 - \sum_{k:k<i} \rho_{kj})(1 - \sum_{k:k\leq i} \rho_{kj})} + \frac{1}{\lambda} \cdot \sum_{j=1}^{N} \sum_{i=1}^{K} c_i \, \lambda_i \, p_{ij} \, x_{ij}.$$

With a change of variables, setting $u_{ij} = \sum_{k=1}^{i} \rho_{kj}$, our original variables p_{ij} can be written in terms of u_{ij} as follows:

$$p_{ij} = \frac{u_{ij} - u_{(i-1)j}}{\lambda_i x_{ij}}.$$

Furthermore, the term $\sum_{i=1}^{K} \frac{c_i \lambda_i p_{ij}}{(1 - \sum_{k:k<i} \rho_{kj})(1 - \sum_{k:k\leq i} \rho_{kj})}$ simplifies to

$$\sum_{i=1}^{K} \frac{1}{1 - u_{ij}} \left(\frac{c_i}{x_{ij}} - \frac{c_{i+1}}{x_{(i+1)j}} \right).$$

Using these simplifications, we can express our objective function $\mathcal{F}$ as a function of the u_{ij} variables.

We then take the Lagrangian of $\mathcal{F}$ by relaxing the "assignment" constraints. Using δ_i to be the multiplier corresponding to the i^{th} customer class, we can write the Lagrangian $\mathbf{L}$ for this problem as

$$L(\mathbf{u}, \delta) = \mathcal{F}(\mathbf{u}) + \sum_{i=1}^{K} \delta_i \sum_{j=1}^{N} \frac{u_{ij} - u_{(i-1)j}}{\lambda_i x_{ij}}.$$

We now differentiate $L(\mathbf{u}, \delta)$ with respect to the new variables u_{ij} and set the derivatives equal to zero. After some algebraic manipulations, we obtain an equation of the form

$$(1 - u_{ij})^2 = r_{ij},$$

where r_{ij} does not depend on u_{ij} and is linear in the other variables δ_i. Such an equation has at most one root in the interval $(0, 1)$. ■

As noted above, Theorem 2.2 significantly simplifies the solution of the nonlinear programming problem (RS). For example, starting from an interior point and applying standard gradient methods, we can find a local minimum of the objective function. If this happens to be in the interior of the domain, then we are done; otherwise, we iterately apply this algorithm starting from all of the lower-dimensional faces. While this property still does not guarantee polynomial-time solvability, it will often lead to efficient solutions for many problems in practice.

2.4 Minimizing Response Time Variance

Let T_i^* denote the mean response time of class i customers under the optimal routing matrix $\mathbf{P}^*$ obtained from the solution of the optimization problem (RS) based on Theorem 2.2, and under the priority rule of Proposition 2.1 at each server. Our analysis in the previous sections establishes that the performance vector $\mathbf{T}^* = (T_1^*, T_2^*, \ldots, T_K^*)$ minimizes the objective function in equation (2). However, there clearly can be multiple scheduling policies that achieve $\mathbf{T}^*$. In this section, we develop a scheduling method that realizes $\mathbf{T}^*$ while tending to also minimize a function of the per-class response time variances.

Squillante and Tsoukatos [24] consider an optimal sequencing strategy for minimizing a function of per-class second moment measures of response time within the context of the multiclass nonpreemptive $M/G/1$ queue of Proposition 2.1, which is formulated as an optimization problem under appropriate constraints and is solved by applying the Kuhn-Tucker Theorem. They show that a structural property of the optimal solution is to equalize a per-class function of the response time for each individual customer, over all customers and all classes. One can then argue, as in [24], that an approach which tends to exhibit this structural property for a particular instance of the objective function is based on the use of general *time-based functions* to control the allocation of resources to classes of customers [8]. Time-function scheduling is in part a generalization of the linear time-dependent priority discipline [11] in which the priority of each customer increases (linearly) according to a per-class function of its time in the system and the customer with the highest instantaneous priority value in the queue is selected for execution at each scheduling epoch. This is based on the observation that, under linear time-dependent priorities, customers tend to be given the server once they reach priority values which are fairly similar across all of the customers. In particular, it can be easily established in the heavy traffic limit as $\widehat{\rho} \rightarrow 1$ that a linear time-function scheduling strategy will satisfy [14]

$$\sigma_k \cdot W_k = \gamma$$

for all classes k, where σ_k is the slope of the class k time-function, $\widehat{\rho}$ is the traffic intensity of the queue, and $\gamma > 0$.

We therefore derive a particular instance of time-function scheduling for the servers that, together with the optimal routing matrix $\mathbf{P}^*$, achieves $\mathbf{T}^*$ and also tends to minimize a function of the per-class response time variances. Consider a stable multiclass nonpreemptive $M/G/1$ queue with class k arrival rate $\widehat{\lambda}_k$ and class k service time distribution $\widehat{G}_k$ having mean $\widehat{x}_k$ and second moment $\widehat{x}_k^{(2)}$; throughout this section we will use the class index k under the constraints $k = 1, 2, \ldots, K$, unless noted otherwise. A linear time-dependent queueing discipline is employed with per-class priority function slopes σ_k such that $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_K$. It can be easily shown that the expected waiting time for customers of class k in this queueing system is given by [11]

$$\widehat{W}_k = \frac{\sum_{i=1}^{K} \frac{\widehat{\lambda}_i \widehat{x}_i^{(2)}}{2(1-\widehat{\rho})} - \sum_{i=k+1}^{K} \widehat{\rho}_i \widehat{W}_i (1 - \sigma_i/\sigma_k)}{1 - \sum_{i=1}^{k-1} \widehat{\rho}_i (1 - \sigma_k/\sigma_i)}, \qquad (7)$$

where $\widehat{\rho}_k = \widehat{\lambda}_k \widehat{x}_k$ and $\widehat{\rho} = \sum_{k=1}^{K} \widehat{\rho}_k$. Note the very simple dependence that $\widehat{W}_k$ has on the slope parameters, namely that these slopes only appear as ratios.

Without loss of generality, let $\sigma_K = 1$. Following [15, 22], we define

$$\alpha_k = \sum_{i=1}^{K} \frac{\widehat{\lambda}_i \widehat{x}_i^{(2)}}{2(1 - \rho)} - \sum_{i=k+1}^{K} \widehat{\rho}_i \, \widehat{W}_i,$$

$$\beta_k = 1 - \sum_{i=1}^{k-1} \widehat{\rho}_i,$$

$$U_k = \sum_{i=k+1}^{K} \widehat{\rho}_i \, \sigma_i \, \widehat{W}_i,$$

$$V_k = \sum_{i=1}^{k-1} \frac{\widehat{\rho}_i}{\sigma_i}.$$

Equation (7) can then be rewritten as

$$\widehat{W}_k = \frac{\alpha_k + U_k/\sigma_k}{\beta_k + \sigma_k V_k},$$

which upon substituting the relations $V_k = V_{k+1} - \widehat{\rho}_k/\sigma_k$ and $\beta_{k+1} = \beta_k - \widehat{\rho}_k$ from the above definitions and simplifying yields

$$\sigma_k = \frac{-(\widehat{W}_k\beta_{k+1} - \alpha_k) \pm \sqrt{(\widehat{W}_k\beta_{k+1} - \alpha_k)^2 + 4U_k\widehat{W}_k V_{k+1}}}{2\widehat{W}_k V_{k+1}},$$

for $k = 1, \ldots, K-1$. Since $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_K = 1$, we have

$$\sigma_k = \frac{-(\widehat{W}_k\beta_{k+1} - \alpha_k) + \sqrt{(\widehat{W}_k\beta_{k+1} - \alpha_k)^2 + 4U_k\widehat{W}_k V_{k+1}}}{2\widehat{W}_k V_{k+1}}, \tag{8}$$

for $k = 1, \ldots, K-1$. This expression can then be used in a straightforward manner together with the above definitions and relations to recursively obtain the values $\sigma_{K-1}, \ldots, \sigma_1$ of the linear time-function policy that will achieve the given performance vector $(\widehat{W}_1, \widehat{W}_2, \ldots, \widehat{W}_K)$.

Hence, by setting $\widehat{\lambda}_k = \lambda_k p^*_{kj}$, $\widehat{x}_k = x_{kj}$, $\widehat{x}^{(2)}_k = x^{(2)}_{kj}$ and $\widehat{W}_k = W^*_{kj}$, where the values of W^*_{ij} corresponding to the optimal performance vector $\mathbf{T}^*$ are known from our analysis in the previous sections (see equation (5)), we have the desired result for the equivalent linear time-function sequencing policy at server j.

3 General Arrival Case

We now remove the assumption of Poisson arrivals by considering a fluid-model approximation of the optimal scheduling problem considered in Section 2. A fluid approximation of a stochastic system is its deterministic, continuous analog that models the asymptotic behavior of the queueing system. We first provide a formal description of the corresponding fluid control problem. Then an analysis of the sequencing and random splitting aspects of the fluid optimization problem is derived respectively in Sections 3.3 and 3.4.

3.1 Fluid Model

Consider the system model of Section 2.1 with the following modifications. Customers arrive and depart in a continuous, deterministic fashion, and thus can be thought of as a *flow of fluid*; in this section we will use the terms "fluid" and "customer" interchangeably. Customers of class i arrive to the system in a continuous manner with rate λ_i, and require x_j time units of processing. A fixed fraction p_{ij} of the class i fluid is routed to server j according to the *routing* matrix $\mathbf{P} \equiv [p_{ij}]_{1 \leq i \leq K; 1 \leq j \leq N}$, independent of all else. The total amount of time required to process one unit of class i fluid on server j is x_i/s_j. For $t \geq 0$, we let $L_{ij}(t)$ be the amount of class i fluid at server j at time t, where the set of initial queue lengths $L_{ij}(0)$ are assumed to be given. Since the quantities $L_{ij}(0)$ will be used often we drop the "time" argument and use L_{ij} instead. Customers of class i incur *holding costs* at rate c_i. In other words, c_i is the cost incurred by a class i customer in the system per unit time. Our scheduling objective now is to minimize the total holding costs incurred until the system empties for the first time, starting from the given initial state L_{ij}. To do this we need to find the routing matrix $\mathbf{P}^*$ and the local sequencing strategies at the servers that result in a schedule of minimum cost.

To put this problem in perspective, it is helpful to make a few remarks about the objective function considered in the fluid and the stochastic systems. In a stochastic system, we typically would like to find a scheduling policy that minimizes a linear function of response times in steady-state. By Little's law, this is equivalent to minimizing a (different) linear function of expected queue lengths in steady-state. In the fluid model, however, we are interested in minimizing the total holding costs incurred until the system empties starting from a specified state, which is related to the queue length processes of the system. Hence, the hope is that the solution of the fluid model, as a function of the initial state, will be useful in determining a near-optimal policy for controlling the stochastic system of interest. For example, fluid-model formulations have been recently used to successfully study general stochastic scheduling problems [4, 1, 13] (different from those considered here).

In the rest of this section we formulate and prove structural properties for the fluid model of the stochastic system considered in Section 2.1. We do not address in this paper the question of how a fluid control can be translated to a control for the stochastic system; refer to [4, 1, 13].

3.2 Formulation

We now can provide a precise formulation of the fluid approximation corresponding to the stochastic system of Section 2. Necessary and sufficient conditions for stability (i.e., the fluid system empties in finite time) are given by equation (1), and thus the system is stable as long as $\rho < S$ which we assume in what follows.

The fluid control problem corresponding to the stochastic system then can be formulated as follows:

$$\begin{aligned}
\text{(CTL)} \quad \min \quad & \int_0^\infty \sum_{i=1}^K \sum_{j=1}^N c_i L_{ij}(t)\, dt \\
\text{s.t.} \quad & \\
& \dot{L}_{ij}(t) = \lambda_i p_{ij} - x_{ij}^{-1} u_{ij}(t), \\
& \sum_{i=1}^K u_{ij}(t) \leq 1, \\
& \mathbf{L}(\mathbf{0}) = \mathbf{given}, \\
& \mathbf{L}(t),\ u_{ij}(t) \geq 0, \quad t \geq 0,
\end{aligned}$$

where $\dot{L}_{ij}(t)$ denotes the derivative of $L_{ij}(t)$ with respect to t.

In our formulation p_{ij} and $u_{ij}(t)$ are the decision variables, where p_{ij} is the fixed fraction of class i fluid routed to server j, and $u_{ij}(t)$ describes the fraction of server j capacity allocated to service class i fluid at time t. Observe that if $L_{ij}(0) = 0$ for all i, j (i.e., if all of the initial queue lengths are zero), the fluid optimal solution will incur zero cost. This is an immediate consequence of the stability condition (1).

3.3 Optimal Sequencing

Let us initially suppose that we have obtained the optimal routing matrix $\mathbf{P}^*$. It follows from the properties of the class of allocation policies considered that the servers are equivalent to multiclass fluid queues. Recall that our interest is in finding the sequencing strategy that provides the globally optimal scheduling solution. Avram, Bertsimas and Ricard [1] showed that the optimal sequencing strategy which minimizes a linear function of expected response times in a multiclass fluid queue is a priority policy, often referred to as the *cμ rule*. Interestingly, the optimal policy is the same for the stochastic system as long as the interarrival times are exponentially distributed. Specifically, they proved the following proposition.

Proposition 3.1 *Consider a multiclass fluid queue with K customer classes. Let λ_i be the class i arrival rate and let x_i be the service time of class i customers, where $\rho_i = \lambda_i x_i$. Let $L_i(t)$ be the amount of class i customers in the system at time t. Then, the scheduling policy Π that gives priority to class i customers over class k whenever $c_i/x_i \geq c_k/x_k$ minimizes $\int_0^\infty \sum_{i=1}^K c_i L_i(t)dt$.*

Observe from Proposition 3.1 that the scheduling policy Π which minimizes the weighted sum of fluid queue lengths is the same for all servers. This follows from the assumption that servers are identical except for their speed and thus no class is given "preferential treatment" by any server. Without loss of generality, assume that customers classes are labeled such that $c_i/x_i \geq c_{i+1}/x_{i+1}$. Under the optimal policy, customer class i is given priority over customer class k if $i < k$.

We now evaluate the cost of class i customers in the multiclass fluid queue of Proposition 3.1. We emphasize that we are looking at a *particular* multiclass fluid queue, namely the one described in Proposition 3.1.

Observe that the fluid system gives priority to class i customers over class k customers if $i < k$. Since the customer arrivals are continuous and deterministic, the fluid system will *always* serve customers. To better understand the sequencing policy we describe it in more detail. Initially the fluid will serve customers of class one. Since the effective arrival rate of class 1 customers is smaller than the rate at which they are served, class 1 customers will eventually deplete; at this point the server (in the fluid system) is ready to serve class two customers. Observe, however, that if the server devotes its full capacity to serve class 2 customers, class 1 customers will start accumulating, and will regain priority. Thus, in an optimal sequencing policy, the server will devote some of its capacity to maintain higher priority classes at zero levels, while working on a lower priority customer class. In our example, the server will devote some capacity to maintain class 1 customers at level zero, and devote the rest of the capacity to class 2 customers. Generalizing this, when the server has depleted all classes up to class $(i-1)$, an appropriate fraction of its capacity will be devoted to keeping all the higher priority customer classes at level zero, and the remaining capacity will be devoted to class i customers. (Contrast this with the optimal sequencing policy in the stochastic system.) Clearly, the amount of "effort" required to maintain a class at level zero can be computed easily: the server has to ensure that such customer classes deplete exactly at the rate at which they arrive.

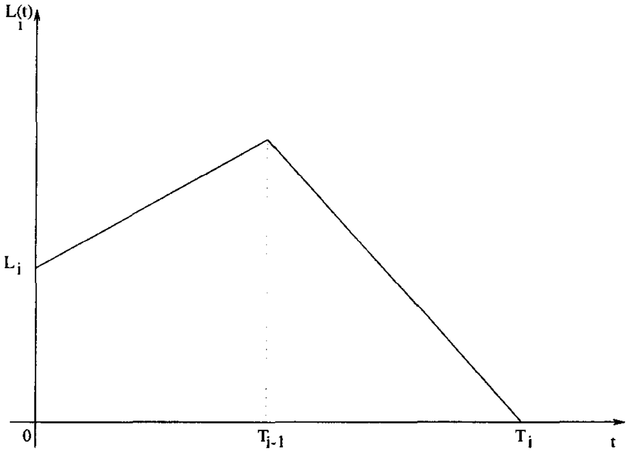

Figure 1: Inventory level of class i

Let T_i be the time at which class i customers deplete from the fluid system, and let $\mu_i = x_i^{-1}$. By the definition of depletion times, and by our ordering of the customer classes, $u_i(t) = 0$ for $t \leq T_{i-1}$, and $u_i(t) = \rho_i$ for $t > T_i$. This immediately shows that the total amount of "effort" required by the server to keep classes one through $(i-1)$ empty is $\sum_{k:k<i}\rho_k$, and thus $u_i(t) = (1 - \sum_{k:k<i}\rho_k)$ for $T_{i-1} < t \leq T_i$. Using the fact that class i customers arrive at rate λ_i, we see that

$$T_i = T_{i-1} + \frac{L_i + T_{i-1}\lambda_i}{\mu_i(1 - \sum_{k:k<i}\rho_k) - \lambda_i}.$$

Solving these sets of linear equations, we obtain

$$T_i = \frac{1}{(1 - \sum_{k:k\leq i}\rho_k)}\left(\sum_{k=1}^{i} L_k x_k\right) \tag{9}$$

Using equation (9) for T_i, we can find the cost incurred by class i customers as follows. The total inventory of class i customers is the area under the curve shown in Figure 1. The total inventory of class i customers is

$$\frac{1}{2}\left\{L_i T_{i-1} + T_i(L_i + \lambda_i T_{i-1})\right\},$$

and thus the total cost of class i customers is given by

$$A_i = c_i \frac{1}{2}\left\{L_i T_{i-1} + T_i(L_i + \lambda_i T_{i-1})\right\},$$

where T_i is given by equation (9).

This simple analysis shows that our optimal scheduling problem is as hard as finding the optimal routing matrix $\mathbf{P}^*$. Notice also that the optimal sequencing policy is identical for all servers and does not depend on $\mathbf{P}^*$. The optimal sequencing policy instead depends only on the weights and the service times of the customer classes. Knowing the optimal sequencing policy enables us to express our objective (the total cost incurred) in terms of $\mathbf{P}^*$. Thus, as in Section 2, the problem of finding the optimal routing matrix can be posed as an (nonlinear) optimization problem, which is considered in the next section.

3.4 Optimal Random Splitting

Based on the above analysis, we formulate the problem of finding an optimal routing matrix as a nonlinear programming problem. Let each server order all of the customers assigned to it according to the priority rule of Proposition 3.1. Since we assume that servers differ only in their speed, the optimal ordering $<_j$ for server j is independent of j and is denoted by $<$. For convenience, and without loss of generality, we assume that the customer classes are relabeled such that

$$\frac{c_1}{x_1} \geq \frac{c_2}{x_2} \geq \frac{c_K}{x_K}.$$

This labeling ensures that the priority ordering of the customer classes at any server is the index order: class i is given priority over class k if $i < k$. Let T_{ij} be the depletion time of class i customers at server j. From our analysis in Section 3.3, we have

$$T_{ij} = \frac{1}{s_j(1 - \sum_{k:k\leq i}\rho_{kj})}\left(\sum_{k=1}^{i} L_{kj} x_k\right), \tag{10}$$

where $\rho_{kj} = \lambda_k p_{kj} x_{kj}$. The total costs incurred by class i customers at server j are then given by

$$A_{ij} = c_i \frac{1}{2}\left\{L_{ij}T_{(i-1)j} + T_{ij}(L_{ij} + \lambda_i p_{ij} T_{(i-1)j})\right\}. \tag{11}$$

Using the analysis and observations in Sections 3.1 and 3.3 together with (1) and (11), our optimal scheduling problem reduces to finding the optimal routing matrix $\mathbf{P}^* \equiv [p^*_{ij}]_{1\le i\le K;1\le j\le N}$ that solves the following optimization problem:

$$\begin{aligned} \text{(FRS)}\quad \min \quad & \sum_{i=1}^{K}\sum_{j=1}^{N} A_{ij} \\ \text{s.t.} \quad & \\ & \sum_{j=1}^{N} p^*_{ij} = 1, \ \forall i, \\ & \sum_{i=1}^{K} \lambda_i x_i p^*_{ij} < \mu_j, \ \forall j, \\ & p^*_{ij} \ge 0, \ \forall i,j. \end{aligned}$$

The optimization problem (FRS) is a nonlinear programming problem and it appears to be difficult to solve in general. However, we establish the following theorem which considerably simplifies the task of solving this problem (for reasons analogous to those given in Section 2.3).

Theorem 3.2 *Any local minimum of problem* (FRS) *in the interior of the feasible domain is a global minimum.*

Proof : The proof follows by an argument identical to the one used in the proof of Theorem 2.2. ■

4 Static Problems

In this section we turn our attention to static instances of the stochastic scheduling problems considered in Sections 2 and 3. When we restrict ourselves to these static cases in which a finite set of customers arrive at time 0 and there are no other arrivals, a formulation based on the above analysis leads us to rederive many results yielding simple approximation algorithms whose guarantees match the best known results. These results have been previously summarized in [18], and they were independently obtained by Skutella in [20].

We first describe the static problems considered and provide an exact nonlinear formulation, and then we present a convex relaxation of this formulation in Section 4.2. We describe a simple (randomized) scheme that rounds a fractional solution of this relaxation to an integer solution in Section 4.3, where we also prove performance guarantees for each of the problems considered. Note that in investigating these static problems, we can restrict ourselves to deterministic scheduling problems without loss of generality; for the static stochastic problems, the processing times are replaced by their expected values, and thus they reduce to the deterministic case.

4.1 Problem description and formulation

We first formulate the scheduling problem under consideration as an integer program. Consider a system with K customers and N servers. Viewing each customer as belonging to its own class, we use i to index the customers, whereas j continues to index the servers. The processing requirement of customer i on server j is z_{ij}; for convenience, define $\mu_{ij} = z_{ij}{}^{-1}$. We emphasize that our formulation is general enough to handle the following three cases of interest:

(a) **Identical servers**: $z_{ij} = z_i$ and is independent of j.

(b) **Uniform servers**: $z_{ij} = z_i/s_j$, where z_i is the processing time requirement of customer i (recall that s_j is the speed of server j).

(c) **Unrelated servers**: z_{ij} depends on both i and j and is an arbitrary positive integer.

For the most part we will work in the setting of *unrelated* servers, as it is the most general case.

Since the problem under consideration is a *static* problem, the allocation question reduces to finding an optimal assignment of customers to servers at time zero. Recall that a positive (integer) weight c_i is associated with customer i and our objective is to minimize $\sum_{i=1}^{K} c_i T_i$, where T_i is the completion time of customer i.

Let p_{ij} be an indicator variable which is one if customer i is assigned to be processed at server j and zero otherwise. Suppose for the moment that we know the optimal assignment of customers to servers — this enables us to reduce the N server scheduling problem to N independent single server scheduling problems. For the objective under consideration, the single server scheduling problem is solved by the $c\mu$ rule [21]: at server j, all the customers assigned to server j are scheduled in such a way that customer k is scheduled before customer ℓ if and only if $c_k\mu_{kj} > c_\ell\mu_{\ell j}$. Motivated by this sequencing policy, we assume that each server orders all the K customers in such a way that customer k appears before customer ℓ in the ordering corresponding to server j (denoted by $k <_j \ell$) if $c_k\mu_{kj} \ge c_\ell\mu_{\ell j}$.

Note that we know the optimal completion time of a customer given an assignment of customers to servers. Thus, we can formulate our problem as that of finding the allocation vector which minimizes weighted completion time. Specfically,

$$\begin{aligned} \min \quad & \sum_{i=1}^{K}\sum_{j=1}^{N} c_i p_{ij} \left(z_{ij} + \sum_{k<_j i} p_{kj} z_{kj} \right) \\ \text{s.t.} \quad & \\ & \sum_{j=1}^{N} p_{ij} = 1, \ \forall i, \\ & p_{ij} \in \{0,1\}, \ \forall i,j. \end{aligned}$$

A few remarks about this formulation are in order. First, observe that an application of the $c\mu$ rule yields

$$T_i = \sum_{j=1}^{N} p_{ij} \left(z_{ij} + \sum_{k<_j i} p_{kj} z_{kj} \right),$$

and thus the optimal solution to our nonlinear integer program does yield an optimal solution to the scheduling problem we started with. Second, this formulation has potentially two complicating factors: (i) the integrality restrictions on the assignment variables; and (ii) the nonlinearity of the objective function. It is quite easy to see that (i) is not a serious problem: a straightforward (randomized) rounding scheme can be used to prove the integrality of the relaxation in which the p_{ij} can assume any value between zero and one (a proof of this is embedded in the proof of Theorem 4.1 in Section 4.3). Hence, the nonlinear optimization problem given by:

$$\begin{aligned} \text{(NLPR)}\quad \min \quad & \sum_{i=1}^{K}\sum_{j=1}^{N} c_i p_{ij} \left(z_{ij} + \sum_{k<_j i} p_{kj} z_{kj} \right) \\ \text{s.t.} \quad & \\ & \sum_{j=1}^{N} p_{ij} = 1, \ \forall i, \\ & p_{ij} \ge 0, \ \forall i,j; \end{aligned}$$

is an exact nonlinear formulation of our scheduling problem in the sense that one can always find an integral optimal solution to this nonlinear programming problem. Although this is an interesting property, it does not simplify our problem because our relaxation is a nonlinear programming problem and thus is difficult to solve. In the next section we will provide a convex relaxation which can be used to design approximation algorithms — of course, the known hardness results of this problem suggest that our convex relaxation is truly a relaxation and does not always provide the optimal solution to the scheduling problems considered.

4.2 Convex relaxations

To clarify the exposition, we fix a server j and assume that our K customers are labeled such that $1 <_j 2 <_j \ldots <_j K$. Under these assumptions, the contribution X_j of server j to the total cost can be written as $X_j = \sum_{i=1}^{K} c_i p_{ij} \left(z_{ij} + \sum_{k<i} p_{kj} z_{kj}\right)$.

When the p_{ij} are restricted to be either zero or one, we have $p_{ij}^2 = p_{ij}$. Simple algebraic manipulations yield

$$\sum_{k=1}^{K}\sum_{\ell=1}^{k-1} c_k p_{kj} z_{\ell j} p_{\ell j} = \sum_{k=1}^{K}\sum_{\ell=k+1}^{K} c_\ell p_{\ell j} z_{kj} p_{kj}. \tag{12}$$

We then can write two expressions for X_j using these observations:

$$\begin{aligned} X_j &= \sum_{k=1}^{K} c_k z_{kj} p_{kj} + \sum_{k=1}^{K}\sum_{\ell=1}^{k-1} c_k p_{kj} z_{\ell j} p_{\ell j}, & (13)\\ &= \sum_{k=1}^{K} c_k z_{kj} p_{kj}^2 + \sum_{k=1}^{K}\sum_{\ell=k+1}^{K} c_\ell p_{\ell j} z_{kj} p_{kj}. & (14) \end{aligned}$$

Adding equations (13) and (14), we have

$$\begin{aligned} X_j = \frac{1}{2}\Bigg(&\sum_{k=1}^{K} c_k z_{kj} p_{kj} + \sum_{k=1}^{K}\sum_{\ell=k+1}^{K} c_\ell z_{kj} p_{kj} p_{\ell j} + \\ &\sum_{k=1}^{K}\sum_{\ell=1}^{k-1} c_k z_{\ell j} p_{kj} p_{\ell j} + \sum_{k=1}^{K} c_k z_{kj} p_{kj}^2\Bigg). \end{aligned} \tag{15}$$

To see why X_j given by equation (15) is convex, we first find the Hessian of X_j. The Hessian of X_j, denoted by $\mathbf{H_j}$, is a $K \times K$ matrix with its $(k,l)^{\text{th}}$ entry given by

$$(\mathbf{H_j})_{k\ell} = \begin{cases} c_\ell z_{kj} & \ell \geq k \\ c_k z_{\ell j} & \ell \leq k \end{cases}$$

To prove that X_j is convex, it suffices to show that $\mathbf{H_j}$ is a positive semidefinite matrix. This is immediate from the ordering of the customers; recall that all customers were ordered so that $c_1\mu_{1j} \geq c_2\mu_{2j} \geq \ldots \geq c_K\mu_{Kj}$, which readily yields a proof of the semidefiniteness of $\mathbf{H_j}$. In fact, if there are no ties, the Hessian is positive definite.

This simple change of using p_{ij}^2 instead of p_{ij} at appropriate places has helped us convert a nonconvex function to a convex one. While this change does not affect the integer program, it results in a weaker relaxation, which fortunately is convex. The convex relaxation of the parallel server scheduling problem we started with is:

$$\begin{aligned} \text{(UPM)}\quad \min\quad & \sum_{j=1}^{N} X_j \\ \text{s.t.}\quad & \\ & X_j \geq \frac{1}{2}\left(\sum_{k=1}^{K} c_k z_{kj}(p_{kj} + p_{kj}^2) + 2\sum_{k=1}^{K}\sum_{l=k+1}^{K} c_l z_{kj} p_{kj} p_{lj}\right), \ \forall j, & (16)\\ & X_j \geq \sum_{i=1}^{K} c_i z_{ij} p_{ij}, \ \forall j, & (17)\\ & \sum_{j=1}^{N} p_{ij} = 1, \ \forall i, \\ & p_{ij} \geq 0, \ \forall i,j. \end{aligned}$$

All of the constraints in the convex relaxation are straightforward and follow from our discussion. We have added equation (17) to the relaxation because it is a linear constraint that certainly yields a lower bound on the optimal value for the integer problem. This constraint does not make a difference to the quality of the relaxation in the case of identical or uniform servers, but it strengthens the relaxation for the case of unrelated servers.

4.3 Rounding

The convex relaxation proposed in the previous section can be solved efficiently using standard techniques. Given a solution to the relaxation, we consider the following straightforward rounding algorithm.

ALGORITHM RR

Step 1: Solve the convex programming problem (UPM), and obtain the optimum routing matrix $\mathbf{P}^*$.

Step 2: Route customer i to server j with probability p_{ij}^*.

Theorem 4.1 *Algorithm RR produces a schedule whose cost is within a factor of $\frac{3}{2}$ of the optimal cost for all three versions of the parallel server scheduling problem.*

Proof: Let the optimum solution to the convex program (UPM) be denoted by $\mathbf{P}^*$ with optimum value $Y^* = \sum_{j=1}^{N} X_j^*$. The integer solution resulting from the randomized rounding algorithm is denoted by $\bar{p}$ with value $\bar{Y} = \sum_{j=1}^{N} \bar{X}_j$. We will analyze the contribution of server j (to the total cost) and show that $E[\bar{X}_j] \leq \frac{3}{2} X_j^*$.

Since we do *independent* rounding, we have

$$E[\bar{p}_{kj}\bar{p}_{lj}] = E[\bar{p}_{kj}]E[\bar{p}_{lj}]. \tag{18}$$

Moreover, the contribution of server j to the total cost can be expressed as

$$\bar{X}_j = \sum_{i=1}^{K} c_i \bar{p}_{ij}(z_{ij} + \sum_{k=1}^{i-1} z_{kj}\bar{p}_{kj}). \tag{19}$$

Using equations (18) and (19), we can compute the expected cost $E[\bar{X}_j]$ due to server j as follows.

$$\begin{aligned} E[\bar{X}_j] &= E\left[\sum_{i=1}^{K} c_i \bar{p}_{ij}(z_{ij} + \sum_{k=1}^{i-1} z_{kj}\bar{p}_{kj})\right] \\ &= \sum_{i=1}^{K} E\left[c_i \bar{p}_{ij}(z_{ij} + \sum_{k=1}^{i-1} z_{kj}\bar{p}_{kj})\right] \\ &= \sum_{i=1}^{K}\left\{E\left[c_i z_{ij}\bar{p}_{ij}\right] + \sum_{k=1}^{i-1} E\left[c_i \bar{p}_{ij}\bar{p}_{kj} z_{kj}\right]\right\} \end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^{K} \left\{ c_i z_{ij} p_{ij}^* + \sum_{k=1}^{i-1} c_i p_{ij}^* p_{kj}^* z_{kj} \right\} \\ &= \sum_{i=1}^{K} c_i z_{ij} p_{ij}^* + \sum_{i=1}^{K} \sum_{k=1}^{i-1} c_i p_{ij}^* p_{kj}^* z_{kj}. \qquad (20) \end{aligned}$$

However, from constraints (16) and (17) we have

$$\begin{aligned} X_j^* &\geq \frac{1}{2} \left(\sum_{k=1}^{K} c_k z_{kj} (p_{kj}^* + (p_{kj}^*)^2) + \right. \\ &\qquad \left. 2 \sum_{k=1}^{K} \sum_{l=k+1}^{K} c_l z_{kj} p_{kj}^* p_{lj}^* \right), \qquad (21) \\ X_j^* &\geq \sum_{i=1}^{K} c_i z_{ij} p_{ij}^*. \qquad (22) \end{aligned}$$

We then obtain from equations (12) and (21)

$$\begin{aligned} X_j^* - \frac{1}{2} \sum_{k=1}^{K} c_k z_{kj} (p_{kj}^*)^2 + \frac{1}{2} \sum_{k=1}^{K} c_k z_{kj} p_{kj}^* &\geq \\ \sum_{i=1}^{K} c_i z_{ij} p_{ij}^* + \sum_{i=1}^{K} \sum_{k=1}^{i-1} c_i p_{ij}^* p_{kj}^* z_{kj}. \qquad (23) \end{aligned}$$

Substituting equation (22) in the left hand side of (23) yields

$$\begin{aligned} X_j^* - \frac{1}{2} \sum_{k=1}^{K} c_k z_{kj} (p_{kj}^*)^2 + \frac{1}{2} X_j^* &\geq \\ \sum_{i=1}^{K} c_i z_{ij} p_{ij}^* + \sum_{i=1}^{K} \sum_{k=1}^{i-1} c_i p_{ij}^* p_{kj}^* z_{kj}. \qquad (24) \end{aligned}$$

The right hand side of equation (24) is simply $E[\bar{X}_j]$. The left hand side of equation (24) simplifies to $\frac{3}{2} X_j^* - \sum_{k=1}^{K} c_k z_{kj} (p_{kj}^*)^2$. Noting that $\sum_{k=1}^{K} c_k z_{kj} (p_{kj}^*)^2$ is a non-negative quantity yields $E[\bar{X}_j] \leq \frac{3}{2} X_j^*$. ■

Remarks

(a) We have described our rounding scheme assuming that we can find an optimal solution to the convex relaxation. In practice, however, we can only find an ϵ-approximate solution. The same rounding scheme can be used with an ϵ-approximate solution while retaining the same performance guarantee.

(b) Our rounding scheme can be derandomized using the method of conditional probabilities. This derandomized rounding algorithm can be coupled with the ϵ-approximate solution to actually find an integer solution with value no worse than 3/2 times the optimum. This is an ϵ-improvement over the previously best known algorithm due to Schulz and Skutella [17].

(c) The argument leading to equation (20) also establishes the integrality of the nonlinear programming relaxation (NLPR).

4.3.1 Identical Servers

For the special case in which all of the servers are identical, the guarantee we can prove is still only 3/2. However, just as in [17], the derandomized version of algorithm RR is exactly the algorithm due to Kawaguchi and Kyan [10], who prove a guarantee of $\frac{\sqrt{2}+1}{2}$ using complicated arguments. Interestingly, for this special case, an optimal solution to the simpler convex relaxation, namely

$$\begin{aligned} \text{(IPM)} \quad \min \quad & \sum_{j=1}^{N} X_j \\ \text{s.t.} \quad & \\ & X_j \geq \frac{1}{2} \left(\sum_{k=1}^{K} c_k z_{kj} (p_{kj} + p_{kj}^2) + \right. \\ & \qquad \left. 2 \sum_{k=1}^{K} \sum_{l=k+1}^{K} c_l z_{kj} p_{kj} p_{lj} \right), \quad \forall j, \\ & \sum_{j=1}^{N} p_{ij} = 1, \quad \forall i, \\ & p_{ij} \geq 0, \quad \forall i, j; \end{aligned}$$

can be computed in closed form: setting $p_{ij} = \frac{1}{N}$ yields an optimal solution to (IPM). Moreover, we can still prove that algorithm RR applied to an optimal solution of (IPM) will yield an integer solution which is no worse than 3/2 times the optimum.

Theorem 4.2 *The convex programming problem* (IPM) *can be solved in closed form for the special case of identical parallel servers. In this case, $\hat{p}_{ij} = \frac{1}{N}$ for all i, j is an optimal solution.*

Proof: We prove this by showing that the solution $\hat{p}_{ij} = \frac{1}{N}$ satisfies the Kuhn-Tucker conditions. (This suffices because our optimization problem is convex.) For convenience, we set $X = \sum_{j=1}^{N} X_j$ and $h_i(p) = \sum_{j=1}^{N} p_{ij} - 1$. It then suffices to show that the set of NK equations

$$\Delta X(\hat{p}) + \sum_{k=1}^{K} u_k \Delta h_k(\hat{p}) = 0$$

can be solved for the u_i. Simple computations show that the $(i, j)^{\text{th}}$ equation in this system (corresponding to the customer i at server j) is

$$c_i z_i + \frac{2 c_i z_i}{N} + \frac{2 c_i \sum_{k=1}^{i-1} z_k}{N} + \frac{2 z_i \sum_{k=i+1}^{K} c_k}{N} + u_i = 0,$$

which is *independent* of j and hence can be solved for u_i. ■

5 Conclusions

In this paper we studied the problem of scheduling different classes of customers on multiple distributed heterogeneous servers to minimize a general objective function based on per-class mean response times. This problem arises in a wide range of distributed computer applications and system environments, as well as communication network environments. We first observed within the context of our model that the optimal sequencing strategy at each of the servers is a simple static priority policy. We then argued based on this observation that the global scheduling problem reduces to finding an optimal routing matrix under this sequencing policy. We formulated the latter problem as a nonlinear programming problem and showed that any interior local minimum is a global minimum, which significantly simplifies the solution of the optimization problem. In the case of independent Poisson arrival streams, we provided an optimal scheduling strategy that also tends to minimize a function of the per-class response time variances. We then considered the case of general arrivals by developing a fluid-model formulation of the optimization problem and deriving an analogous set of results. Applying our analysis to various static instances of the general problem led us to rederive many results yielding simple approximation algorithms whose guarantees match the best known results.

References

[1] F. Avram, D. Bertsimas, and M. Ricard. Fluid models of sequencing problems in open queueing networks; an optimal control approach. In F. Kelly and R. Williams, editors, *Stochastic Networks*, volume IMA 71, pages 199–234, 1995.

[2] F. Bonomi and A. Kumar. Adaptive optimal load balancing in a nonhomogeneous multiserver system with a central job scheduler. *IEEE Transactions on Computers*, 39(10):1232–1250, October 1990.

[3] S. C. Borst. Optimal probabilistic allocation of customer types to servers. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 116–125, 1995.

[4] H. Chen and D. D. Yao. Dynamic scheduling of a multiclass fluid network. *Operations Research*, 41(6):1104–1115, 1993.

[5] M. E. Crovella, M. Harchol-Balter, and C. Murta. Task assignment in distributed systems: Improving performance by unbalancing load. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 268–269, June 1998.

[6] D. Dias, W. Kish, R. Mukherjee, and R. Tewari. A scalable and highly available Web server. In *Proceedings of the 1996 IEEE Computer Conference (COMPCON)*, February 1996.

[7] D. L. Eager, E. D. Lazowska, and J. Zahorjan. Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, SE-12(5):662–675, May 1986.

[8] L. L. Fong and M. S. Squillante. Time-Function Scheduling: A general approach to controllable resource management. In *Proceedings of the Symposium on Operating Systems Principles (SOSP)*, page 230, December 1995.

[9] G. Hunt, G. Goldszmidt, R. King, and R. Mukherjee. Network dispatcher: A connection router for scalable Internet services. In *Proceedings of the 7th International World Wide Web Conference*, April 1998.

[10] T. Kawaguchi and S. Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4):1119–1129, 1986.

[11] L. Kleinrock. *Queueing Systems Volume II: Computer Applications*. John Wiley and Sons, 1976.

[12] M. Livny and M. Melman. Load balancing in homogeneous broadcast distributed systems. In *Proceedings of the ACM Computer Network Performance Symposium*, pages 47–55, 1982.

[13] C. Maglaras. A methodology for dynamic control policy design for stochastic processing networks via fluid models. In *Proceedings of IEEE Conference on Decision and Control*, 1997.

[14] R. D. Nelson. Heavy traffic response times for a priority queue with linear priorities. *Operations Research*, 38(3):560–563, 1990.

[15] R. D. Nelson. Invertible mapping of waiting times in a M/G/1 queue with linear priorities. Unpublished Draft, June 1993.

[16] K. W. Ross and D. D. Yao. Optimal load balancing and scheduling in a distributed computer system. *Journal of the ACM*, 38(3):676–690, July 1991.

[17] A. Schulz and M. Skutella. Random-based scheduling. Technical Report 549/1997, Department of Mathematics, TU Berlin, 1997.

[18] J. Sethuraman and M. S. Squillante. Optimal scheduling of multiclass parallel machines. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1999.

[19] K. Sevcik. A proof of the optimality of 'smallest rank' scheduling. *Journal of the ACM*, 21:66–75, 1974.

[20] M. Skutella. Semidefinite relaxations for parallel machine scheduling. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, November 1998.

[21] W. E. Smith. Various optimizers for single-stage production. *Naval Research and Logistics Quarterly*, 3:59–66, 1954.

[22] M. S. Squillante, L. L. Fong, S. Liu, and S. K. Ryan. A control study of time-function scheduling: Part I. Technical Report RC 19765, IBM Research Division, September 1994.

[23] M. S. Squillante and R. D. Nelson. Analysis of task migration in shared-memory multiprocessors. In *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pages 143–155, May 1991.

[24] M. S. Squillante and K. P. Tsoukatos. Fundamentals of time-function scheduling. Unpublished Draft, September 1995.

[25] A. N. Tantawi and D. Towsley. Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32(2):445–465, April 1985.

[26] Y. T. Wang and R. Morris. Load sharing in distributed systems. *IEEE Transactions on Computers*, C-34(3):204–217, 1985.