Performance Evaluation 68 (2011) 414–428

Contents lists available at ScienceDirect

Performance Evaluation

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

Optimal resource allocation for time-reservation systems

Ran Yang [a,b,*], Sandjai Bhulai [a,b], Rob van der Mei [b,a], Frank Seinstra [a]

[a] VU University Amsterdam, Faculty of Sciences, De Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands
[b] CWI, Advanced Communication Networks, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 16 February 2010
Received in revised form 1 December 2010
Accepted 12 January 2011
Available online 25 January 2011

Keywords:
Constrained Markov decision problems
Monotonicity
Optimal resource allocation
Queueing theory
Time-reservation systems

ABSTRACT

This paper studies the optimal resource allocation in time-reservation systems. Customers arrive at a service facility and receive service in two steps; in the first step information is gathered from the customer, which is then sent to a pool of computing resources, and in the second step the information is processed after which the customer leaves the system. A central decision maker has to decide when to reserve computing power from the pool of resources, such that the customer does not have to wait for the start of the second service step and that the processing capacity is not wasted due to the customer still being serviced at the first step. The decision maker simultaneously has to decide on how many processors to allocate for the second processing step such that reservation and holding costs are minimized. Since an exact analysis of the system is difficult, we decompose the system into two parts which are solved sequentially leading to nearly optimal solutions. We show via dynamic programming that the near-optimal number of processors follows a step function with as an extreme policy the bang–bang control. Moreover, we provide new fundamental insights in the dependence of the near-optimal policy on the distribution of the information gathering times. Numerical experiments demonstrate that the near-optimal policy closely matches the performance of the optimal policy of the original problem.

1. Introduction

In recent years new real-time multimedia services have triggered a tremendous growth in data volumes and computational demands. Typical services include iris-scan and fingerprint systems, that make high-resolution scans and require processing of the data to identify a person; these services operate in a real-time environment and run under very strict time constraints. To adhere to such constraints, these large-scale services typically use centralized computing clusters to execute on. In large-scale systems, the applications can reserve a number of processing resources to process the data. This gives rise to a new class of models in which the application has to decide *when* to reserve the processing resources and *how many*. In this decision making, there is a trade-off between the lead time on one hand and the operating costs on the other hand. Making a reservation too early leads to inefficiency, since the processing resources have to wait on the data gathering/scanning process; a too late reservation leads to unnecessarily long customer waiting times. Allocating too many processing resources results in a short lead time, but comes at high allocation costs; allocating too few resources leads to long processing times and blocks computing resources for the next customer.

In the literature, a lot of research has been devoted to resource allocation problems. In the context of protocol design, Daniel and Chronopoulos [1] have investigated the problem for resource allocation involving selfish agents, and designed a truthful mechanism for solving the load-balancing problem in heterogeneous distributed systems. Park [2] present a scalable

* Corresponding author at: VU University Amsterdam, Faculty of Sciences, De Boelelaan 1081a, 1081 HV, Amsterdam, The Netherlands.
E-mail address: ryang@few.vu.nl (R. Yang).

0166-5316/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2011.01.003

protocol for fast co-allocation of Internet resources that ensures deadlock and livelock freedom during the resource co-allocation process. Rana et al. [3] focus on the modeling and detection of conflicts that arise during resource discovery and application scheduling. They propose an approach that helps to resolve conflicts and enables each resource and application to respond to changes in the environment.

In a Grid computing environment, there are several papers that study architectures that enhance resources with online control, online monitoring, and decision procedures. Czajkowski et al. [4] develop implementations of two co-allocation strategies in the context of the Globus toolkit [5]. In one strategy, all the required resources are specified at the time the request is made. The request succeeds if all resources required are allocated, and otherwise, the request fails and none of the resources is acquired. The other strategy allows for application-level guidance of resource selection and failure handling prior to commitment. Foster et al. [6] describe an implementation of a mechanism that enables the coordinated use of reservation and adaptation within the GARA resource management architecture [7]. Furthermore, they develop three application-level adaptive control mechanisms: two that use loss-rate information to adapt reservations and one that uses reservation state information to adapt the transmission rate. Wang and Luo [8] set up the layered structure of Grid QoS, which provides a reasonable gist for mapping and converting QoS parameters in grids so that it can implement the user's QoS requirements in the process of grid resource allocation management.

Other research has focused on economic models in a Grid computing environment. Sandholm et al. [9] develop a suite of prediction models and tools to aid the users in deciding how much funding their jobs would need to complete within a certain deadline, or conversely, when a job would be expected to complete given a budget. In this work, one tries to find the best bidding strategy based on the so-called Best Response optimization algorithm (see [10]) that aims to maximize the utility of users across a set of resources, under the constraint that the total budget of the user is given. Buyya et al. [11] give an overview of different economic models for resource trading and establishing pricing strategies, and discuss the resource trading in Grid brokering and offer an infrastructure for resource management and trading in the Grid environment (see, e.g., [12–15] for more details).

The works on cost/reservation optimization in the context of resource allocation that are closest to our model are [16–19]. Aziz and El-Rewini [16] present a framework for resource allocation and task scheduling, where the objective function is to minimize the job completion time and to minimize the number of resources needed for the completion of the job. Nurmi et al. [17] propose a statistical method, called VARQ [20], for job scheduling. Using QBETS [18] to compute a time bound on the delay a specific user job will experience, VARQ implements a reservation by determining when a job should be submitted to a batch queue to ensure that it will be running at a particular time in the future. Fukuda et al. [19] study the relationship between the video quality and the required number of CPUs and network resources to provide a real-time video presentation. Based on this relationship, they propose a resource-allocation scheme to share resources fairly among users by solving the utility-maximization problem, where the utility is a function of the video quality and the resource allocation costs.

The main difference between the existing literature and our work is that we aim to (1) optimize the resource-allocation costs and reservation moments *simultaneously* and (2) satisfy a QoS constraint on the delay of a job. This is in contrast to the aforementioned works which only have a focus on optimizing either the resource allocation costs [16,19] or the reservation moments [18,17]. Moreover, none of these works provides a QoS guarantee on the sojourn time of a job in the system.

More specifically, we study the optimal resource allocation in time-reservation systems. Customers arrive at a service facility and receive service in two steps: an information *gathering* and an information *processing* step. The system has to decide when and how much computing power to reserve for the second step such that the allocation costs are minimized while satisfying a response-time constraint. Since an exact analysis of the system is difficult, we decompose the system into two parts which are solved sequentially leading to an approximation of the original problem posed. We show via monotonicity of the dynamic programming optimality operator that the optimal number of processors for the decomposed problem follows a step function with as extreme policy the bang–bang control (see [21] for a discussion on the bang–bang control). Furthermore, we show how the optimal policy in the decomposed problem varies when the distributions of the reservation times and information gathering times, and the response-time constraint are varied.

The rest of the paper is organized as follows. In Section 2 we formulate the model. Next, we derive the structure of the optimal policy optimal policy in the decomposed problem in Section 3. In Section 4 we illustrate these results by numerical experiments and study the impact of variability under different service distributions. Finally, in Section 5 we end with conclusions and discuss topics for further research.

2. Model formulation

Consider a service facility at which customers arrive according to a Poisson process with rate λ. The service that the facility provides to its customers consists of two servicing steps (see Fig. 1): first, a customer receives service at a service station 1 that gathers and pre-processes data to be used in the next service step. Then, the data is transferred to a pool of computing resources, service station 2, where the second service step takes place. After a customer has received the second service step, he leaves the system. The service facility is subject to the service level constraint stating that $\mathbb{E}S \leq \alpha$, where the sojourn time S is defined as the time from the arrival of the customer until the departure of the customer.

We model service station 1 as a single-server station with an infinite buffer queue. A newly arriving customer receives service immediately when upon arrival the station is not occupied, and he waits in the queue otherwise. The system incurs

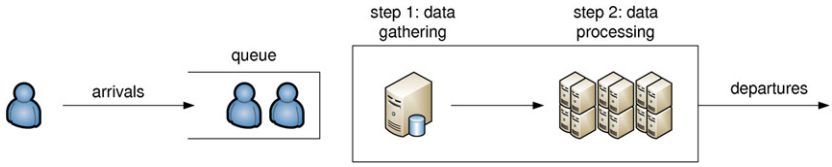


Fig. 1. Service facility with two processing steps.

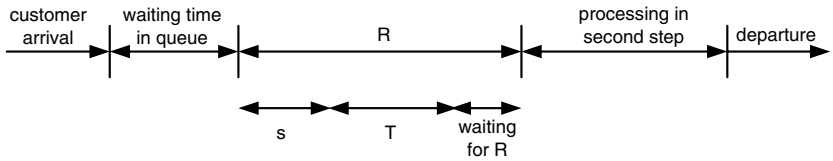


Fig. 2. Computing resources are available before end of service at step 1.

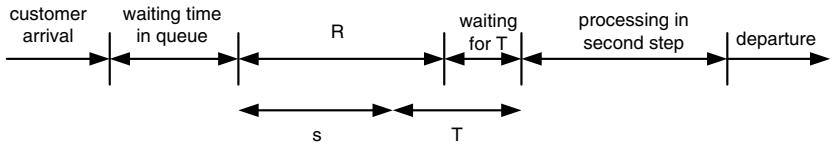


Fig. 3. The service at step 1 is finished while computing resources are not available.

waiting costs $c_1(x)$ per time unit for having x customers in the system, with $c_1(x)$ an increasing function in x. The service duration at the first step is modeled by the random variable R. The time to set up computing resources to process the data is modeled by the random variable T (independent of R and independent of the number of allocated computing resources [22]). The facility can choose on the start of the service when to make a reservation for these computing resources, i.e., it wants to select the moment $s \geq 0$ at which to start the reservation after starting the service. Note that if the facility reserves the computing resources too early, then unnecessary computing resources are blocked and remain idle. Fig. 2 represents exactly this case from the viewpoint of the customer in iris-scan and fingerprint systems. The figure shows all the steps a customer undergoes in the system; first, a customer arrival occurs after which there is a potential waiting time in the queue before he reaches the server at step 1. Upon the start of his service at step 1, a decision s is made for reserving the computing resources. This results in two competing processes; the service process R and the reservation process $s+T$. When *both* have completed, the customer moves on to the service in step 2, while still blocking the server in step 1 (there can only be at most one customer in service in both steps). After the service at step 2, the customer departs the system and a new customer can be served at step 1. Recall that the sojourn time S is defined as the time from the arrival of the customer until the departure of the customer. On the other hand, if the resources are reserved too late, then the customer has to wait before the system can process his data leading to high waiting costs. This situation is reflected in Fig. 3. The aim in this service step is to balance the end of the service time and the moment that the resources are reserved and available, e.g., the system tries to choose s such that $C(s) := \mathbb{E}(R - s - T)^2$ is minimized.

After having received service at service station 1, the customer remains in front of service station 1, but now receives service from service station 2. Service station 2 is a pool of computing resources consisting of A identical parallel processors (e.g., the DAS-3 environment [23]). When the reservation time of the computing resources has finished, the customer can be served at service station 2. The service facility can choose how many processors to use for the processing of the data. We assume that there are costs $c_2(a)$ for using a processors per time unit, with $c_2(a)$ an increasing function in a. Clearly, having too few processors leads to a violation of the service level, and having too many processors leads to high reservation costs. We assume that the service follows an exponential distribution with service rate $\mu(a)$ when a processors are used, with μ an increasing function in a. In the ideal case one would have $\mu(a) = \mu a$, with a fixed service rate μ, however, due to communication overhead the function is sublinear in practice. Moreover, we suppose that A is large enough so that for the optimal choice of s and a, say (s^*, a^*), there is a policy that meets the service level α.

So far, we have not made any assumptions on the parameters of the system. However, to ensure a stable system one needs to add conditions that take into account both the arrival rate of customers as well as the mean service time of a customer in the system. Note that the latter depends heavily on the policy (s^*, a^*) in use for every state. Therefore, it is difficult to state general stability conditions. One can note, though, that if $(s^*, a^*) = (0, A)$ for all states, then the mean service time is minimized; the mean service time is then given by $\mathbb{E}\max\{R, T\} + 1/\mu(A)$. Hence, if $\lambda\left[\mathbb{E}\max\{R, T\} + 1/\mu(A)\right] < 1$, then there exists at least one policy for which the system is stable.

Problem formulation

The objective is to find a policy that selects (s^*, a^*) based on the number of customers x in the system such that *both* $C(s)$ and the long-term average costs (i.e., waiting plus reservation costs) are minimized while at the same time the service level

constraint α is met. To formulate this objective in mathematical terms, the following notation is useful. Let X_t and A_t denote the random variables denoting the number of customers in the system and the number of reserved processors at time t, respectively. Let π denote a state-dependent policy, then, the optimization problem can be stated as

$$\begin{aligned} &\min_{\pi}\left[C(s) + \mathbb{E}_{\pi}[c_1(X_\infty) + c_2(A_\infty)]\right] \\ &\text{subject to} \quad \mathbb{E}_{\pi}[S] \leq \alpha, \end{aligned} \tag{1}$$

where $C(s)$ is given by $\mathbb{E}(R - s - T)^2$, X_∞ and A_∞ denote the steady-state numbers of the variables X_t and A_t, and where S is the steady-state sojourn time of an arbitrary customer.

Solution approach

The service steps in the above mentioned problem have a mixture of continuous and discrete decision variables with intricate dependencies such that a classical decision theoretic approach is not computationally tractable in general. Therefore, we approximate the optimal policy by following a two-step approach: First, we solve for s^* minimizing $C(s)$ in the first service step. Second, given s^*, we find a^* that minimizes $\left[\mathbb{E}_{(s^*,a)}[c_1(X_\infty) + c_2(A_\infty)]\right]$ subject to $\mathbb{E}_{(s^*,a)}[S] \leq \alpha$ under assumptions provided in Section 3. In Section 4 we show that the resulting approximate policy is nearly optimal in the numerical experiments.

3. Structural properties of the decomposed model

In order to derive an optimal policy (s^*, a^*) we assume that the optimal choice for s does not depend on the policy for the number of servers. Note that this assumption allows us to decompose the problem into two parts, whereby the optimal reservation moment s^* can be studied independently of the optimal allocation a^*. Then, given the optimal parameter s^*, one can study the optimal reservation policy at service station 2. Also note that this assumption is not true in general in the original problem, e.g., when $c_1(x)$ and $c_2(a)$ are really large relative to $C(s)$, then one would choose $s = 0$, whereas $s > 0$ in our decomposed problem. However, in Section 4 we compare our decomposition technique with the optimal policy in the original problem, and show that for realistic parameter values the decomposition technique is close to optimal.

We first start with the optimal reservation moment s^*.

Lemma 3.1. *Let R and T have an arbitrary distribution. Then $C(s) = \mathbb{E}(R - s - T)^2$ is minimized by $s^* = [\mathbb{E}R - \mathbb{E}T]^+$ resulting in $C(s^*) = \mathbb{V}\mathrm{ar}\,R + \mathbb{V}\mathrm{ar}\,T$ for $\mathbb{E}R \geq \mathbb{E}T$, and in $C(s^*) = \mathbb{V}\mathrm{ar}\,R + \mathbb{V}\mathrm{ar}\,T + (\mathbb{E}R - \mathbb{E}T)^2$ for $\mathbb{E}R < \mathbb{E}T$.*

Proof. Let R and T have an arbitrary distribution. Then,

$$\begin{aligned} \mathbb{E}(R - s - T)^2 &= \mathbb{E}(R^2 + s^2 + T^2 - 2sR - 2RT + 2sT) \\ &= \mathbb{E}R^2 + \mathbb{E}T^2 - 2\mathbb{E}R\mathbb{E}T + s^2 - 2s\mathbb{E}R + 2s\mathbb{E}T \\ &= \mathbb{V}\mathrm{ar}\,R + (\mathbb{E}R)^2 + \mathbb{V}\mathrm{ar}\,T + (\mathbb{E}T)^2 - 2\mathbb{E}R\mathbb{E}T + s^2 - 2s(\mathbb{E}R - \mathbb{E}T) \\ &= \mathbb{V}\mathrm{ar}\,R + \mathbb{V}\mathrm{ar}\,T + (\mathbb{E}R - \mathbb{E}T)^2 + [s - (\mathbb{E}R - \mathbb{E}T)]^2 - (\mathbb{E}R - \mathbb{E}T)^2 \\ &= \mathbb{V}\mathrm{ar}\,R + \mathbb{V}\mathrm{ar}\,T + [s - (\mathbb{E}R - \mathbb{E}T)]^2. \end{aligned}$$

From this expression it follows that $C(s)$ is minimized by taking s equal to $\mathbb{E}R - \mathbb{E}T$. Since s is only allowed to be non-negative, $s^* = [\mathbb{E}R - \mathbb{E}T]^+$, with $[z]^+ = \max\{z, 0\}$. The value of $C(s^*)$ then readily follows by substitution. □

In the special case that R and T follow an exponential distribution with parameter δ and γ, respectively, we have that the optimal reservation time is given by

$$s^* = \arg\min_{s} \mathbb{E}(R - T - s)^2 = \left[\frac{\gamma - \delta}{\delta\gamma}\right]^+. \tag{2}$$

In the case of $\gamma > \delta$, i.e., the average reservation time is smaller than the average service time, we have

$$C(s^*) = \mathbb{E}(R - T - s^*)^2 = \frac{\gamma^2 + \delta^2}{\delta^2\gamma^2} = \frac{1}{\delta^2} + \frac{1}{\gamma^2}. \tag{3}$$

Otherwise, we have

$$C(s^*) = \mathbb{E}(R - T)^2 = \mathbb{E}(R^2) + \mathbb{E}(T^2) - 2\mathbb{E}(R)\mathbb{E}(T) = \frac{2}{\delta^2} + \frac{2}{\gamma^2} - \frac{2}{\delta\gamma}. \tag{4}$$

Note that Eq. (3) implies that if $\gamma \gg \delta$ or $\mathbb{E}[R] \gg \mathbb{E}[T]$, then the $\mathbb{E}R$ dominates in the expression of $C(s)$. Hence, relatively, the $\mathbb{E}T$ is not important, thus approximately, it is as good as to reserve the computing nodes directly after the scanning is completed. In this case, $s^* \approx \frac{1}{\delta}$. Eq. (2) implies that if $\gamma < \delta$ or $\mathbb{E}[R] < \mathbb{E}[T]$, then it is optimal to reserve the computing

nodes directly after the scanning starts, thus $s^* = 0$. Note that the expected time between the start of service at station 1 and station 2, given by $\beta := \mathbb{E}\max\{R, s^* + T\}$, is given by $\beta = s^* + \frac{1}{\gamma} + e^{-\delta s^*}\left[\frac{1}{\delta} - \frac{1}{\delta+\gamma}\right]$.

Note that action s^* constitutes a stationary policy for all customers. In general, the arrival process to station 2 is *not* the original Poisson process delayed by the constant β. However, for the purpose of model tractability, we assume that this does hold. To meet the service level that was specified as the requirement that the mean sojourn time of a customer does not exceed α, the mean sojourn time at station 2 is allowed to be at most $\alpha' := \alpha - \beta$ time units. In Section 4 we will show that the error on the mean sojourn time in the system by assuming a Poisson process is not too big.

The assumption that the output process of station 1 is a Poisson process makes the derivation of the optimal resource allocation policy for station 2, i.e., how many processors should be assigned to a customer, tractable. For this purpose, we formulate the problem as a constrained Markov decision problem. Thus, we try to minimize the holding and processing costs with the additional constraint that the mean sojourn time is below α'. The constrained Markov decision problem is, in general, hard to solve. Therefore, we first study the unconstrained problem in which we drop the sojourn time constraint. For this system, we show that the optimal policy possesses a threshold-type structure. We show that the constrained problem possesses a similar structure as the unconstrained problem. Therefore, the structure of the optimal policy of the unconstrained case carries over to the constrained problem, however, we will show that the optimal policy is not deterministic but randomized.

3.1. Unconstrained Markov decision problem

Let $\mathcal{X} = \mathbb{N}_0 = \{0, 1, 2, \ldots\}$ denote the state space, where $x \in \mathcal{X}$ denotes the number of customers present at station 2. For each customer the set of actions is given by $\mathcal{A} = \{0, 1, \ldots, A\}$, where $a \in \mathcal{A}$ denotes the number of processors that is allocated to a customer. When action a is chosen in state x, there are two possible events in the system. First, an arrival of a customer can occur with rate λ. Second, since at any moment in time there can only be one customer in service, the customer finishes his service with rate $\mu(a)$. Note that if multiple customers are allowed to be served simultaneously, then the allocation policy for each customer needs to be kept in the state description as well since every customer is served with a different rate.

Next, we uniformize the system (see Section 11.5 of Puterman [24]). To this end, we assume that the uniformization constant $\lambda + \mu(A) = 1$; we can always get this by scaling. Uniformizing is equivalent to adding dummy transitions (from a state to itself) such that the rate out of each state is equal to 1; then we can consider the rates to be transition probabilities. Thus, the transition probabilities p, are given by $p(x, a, x + 1) = \lambda$ and $p\big(x, a, [x - 1]^+\big) = \mu(a)$. Similarly, the system is subject to costs c consisting of holding costs $c_1(x)$ and costs for the resource allocation $c_2(a)$, thus $c(x, a) = c_1(x) + c_2(a)$. The tuple $(\mathcal{X}, \mathcal{A}, p, c)$ defines the Markov decision problem.

Define a deterministic policy π as a function from $\mathcal{X}$ to $\mathcal{A}$, i.e., $\pi(x) \in \mathcal{A}$ for all $x \in \mathcal{X}$. Let $u_t^\pi(x)$ denote the total expected costs up to time t when the system starts in state x under policy π. Note that for any stable and work-conserving policy, the Markov chain satisfies the unichain condition, so that the average expected costs $g(\pi) = \lim_{t\to\infty} u_t^\pi(x)/t$ is independent of the initial state x (see Proposition 8.2.1 of Puterman [24]). The goal is to find a policy π^* that minimizes the long-term average costs, thus $g = \min_\pi g(\pi)$.

Let $V(x)$ be a real-valued function defined on the state space. This function will play the role of the relative value function, i.e., the asymptotic difference in total costs that results from starting the process in state x instead of some reference state. The long-term average optimal actions are a solution of the optimality equation (in vector notation) $g + V = TV$, where T is the dynamic programming operator acting on V defined as follows:

$$TV(x) = \lambda V(x + 1) + c_1(x) + \min_{a\in\mathcal{A}}\big\{\mu(a)V\big([x - 1]^+\big) + \big(\mu(A) - \mu(a)\big)V(x) + c_2(a)\big\}. \tag{5}$$

The first term in the expression $TV(x)$ models the arrivals of customers to station 2. The second term denotes the holding costs. The third term denotes the departure of a customer in case action a has been chosen. The fourth term is the uniformization constant. The last term models the costs for the resource allocation. The optimality equation $g + V = TV$ is hard to solve analytically in practice. Alternatively, the optimal actions can also be obtained by recursively defining $V_{l+1} = TV_l$ for arbitrary V_0. For $l \to \infty$, the maximizing actions converge to the optimal ones (for existence and convergence of solutions and optimal policies we refer to Chapter 8 of Puterman [24]). This procedure is also called value iteration.

Note that the costs $c_1(x)$ and $c_2(a)$ model the trade-off between the server allocation and the holding costs. The first cost function drives the decision maker to be liberal with the resources, since a too low resource allocation leads to long queues and thus high holding costs. The second cost function ensures that the decision maker is not too liberal since a too high resource allocation leads to high processor costs. This is illustrated by the following two cases.

No processing costs: First, we take $c_2(a)$ to be zero for all $a \in \mathcal{A}$. From Expression (5) we can see that the optimal policy evidently uses all processors (also in the original system), since this maximizes the service rate and minimizes the queue length (since $c_2(a) = 0$). Therefore $a^* = A$.

No holding costs: Next, we assume that $c_1(x) = 0$ for all $x \in \mathcal{X}$. In this case, Expression (5) shows that the emphasis is on processor utilization. Therefore, the optimal action would be to use no processors at all. In practice, this is not a permissible

action, since no services would be provided at all. If the constraint on the sojourn time is taken into account, then the optimal action changes. This will be illustrated in Section 3.2.

We now proceed to the general solution of the unconstrained Markov decision problem by deriving properties of the optimal dynamic policy. To this end, consider the backward recursion operator V_{n+1} given by

$$V_{n+1}(x) = \lambda V_n(x+1) + \min_{a\in\mathcal{A}}\{\mu(a)V_n([x-1]^+) + (\mu(A)-\mu(a))V_n(x) + c_2(a)\} + c_1(x). \tag{6}$$

Using the backward recursion operator we can derive structural properties of the optimal policy. The following result will play an important role for the derivation of the structure of the optimal policy.

Lemma 3.2. *Assume that $c_1(x)$ is a convex increasing function in x and that $c_2(a)$ is an increasing function of a. Then $V(x)$ is a convex increasing function in x.*

Proof. The proof is by induction on n. Let $V_0(x) = 0$ for all x. Then clearly, V_0 is a convex increasing function in x. Now, assume that the statement holds for k, then we prove that the statement also holds for $k+1$. To this end, define

$$T_a^k(x) = \mu(a)V_k([x-1]^+) + [\mu(A)-\mu(a)]V_k(x) + c_2(a).$$

Then, for $x \geq 0$, we have

$$\begin{aligned} V_{k+1}(x+1) - V_{k+1}(x) &= [\lambda V_k(x+2) - \lambda V_k(x+1)] + [c_1(x+1) - c_1(x)] + [\min_a\{T_a^k(x+1)\} - \min_a\{T_a^k(x)\}] \\ &\geq \min_a\{T_a^k(x+1)\} - \min_a\{T_a^k(x)\}. \end{aligned}$$

The inequality holds because the first two terms are non-negative due to the induction hypothesis, and the second two terms follow by the assumption on the cost function c_1. Let $a^* = \arg\min_a\{T_a^k(x+1)\}$, then, for $x \geq 0$, we have

$$\begin{aligned} V_{k+1}(x+1) - V_{k+1}(x) &\geq T_{a^*}^k(x+1) - \min\{T_a^k(x)\} \\ &\geq T_{a^*}^k(x+1) - T_{a^*}^k(x) \\ &= \mu(a^*)V_k(x) + [\mu(A)-\mu(a^*)]V_k(x+1) + c_2(a^*) \\ &\quad - \mu(a^*)V_k([x-1]^+) - [\mu(A)-\mu(a^*)]V_k(x) - c_2(a^*) \\ &\geq 0. \end{aligned}$$

Therefore, by induction, we derive that $V(x+1) - V(x) \geq 0$. Now we proceed to prove convexity of the relative value function. Assume that convexity holds for k, then we need to prove convexity for $k+1$. Then, for $x \geq 0$, we have

$$\begin{aligned} V_{k+1}(x+1) - 2V_{k+1}(x) + V_{k+1}(x-1) &= [\lambda V_k(x+2) - 2\lambda V_k(x+1) + \lambda V_k(x)] + [c_1(x+1) - 2c_1(x) \\ &\quad + c_1(x-1)] + [\min_a\{T_a^k(x+1)\} - 2\min_a\{T_a^k(x)\} + \min_a\{T_a^k(x-1)\}] \\ &\geq \min_a\{T_a^k(x+1)\} - 2\min_a\{T_a^k(x)\} + \min_a\{T_a^k(x-1)\}. \end{aligned}$$

The inequality holds because the first expression between the brackets is non-negative due to the induction hypothesis, and the second from the assumption on c_1. Now assume that $a_1^* = \arg\min_a\{T_a^k(x+1)\}$ and $a_2^* = \arg\min_a\{T_a^k(x-1)\}$. Then, we have

$$\begin{aligned} &\min_a\{T_a^k(x+1)\} - 2\min_a\{T_a^k(x)\} + \min_a\{T_a^k(x-1)\} \\ &\geq [c_2(a_1^*) - c_2(a_1^*) - c_2(a_2^*) + c_2(a_2^*)] + [\mu(a_1^*)V_k(x) - \mu(a_1^*)V_k([x-1]^+) - \mu(a_2^*)V_k([x-1]^+) \\ &\quad + \mu(a_2^*)V_k([x-2]^+)] + [(\mu(A)-\mu(a_1^*))V_k(x+1) - (\mu(A)-\mu(a_1^*))V_k(x) - (\mu(A)-\mu(a_2^*))V_k(x) \\ &\quad + (\mu(A)-\mu(a_2^*))V_k([x-1]^+)] \\ &= (\mu(a_1^*)-\mu(a_2^*))[V_k(x) - V_k([x-1]^+)] + \mu(a_2^*)[V_k(x) - 2V_k([x-1]^+) + V_k([x-2]^+)] \\ &\quad + (\mu(A)-\mu(a_1^*))[V_k(x+1) - V_k(x)] - [\mu(A)-\mu(a_1^*) + (\mu(a_1^*)-\mu(a_2^*))](V_k(x) - V_k([x-1]^+)) \\ &= \mu(a_2^*)[V_k(x) - 2V_k([x-1]^+) + V_k([x-2]^+)] + (\mu(A)-\mu(a_1^*))[V_k(x+1) - 2V_k(x) + V_k([x-1]^+)] \\ &\geq 0. \end{aligned}$$

The first inequality follows from taking a potentially suboptimal action in the second term of $\min_a\{T_a^k(x+1)\} - 2\min_a\{T_a^k(x)\} + \min_a\{T_a^k(x-1)\}$. The two equalities follow by rearranging the terms. The last inequality follows by the induction hypothesis and by noting that $\mu(A) - \mu(a_1^*)$ is positive. Hence, using mathematical induction we have proved that $V(x)$ is a convex increasing function in x. □

The previous lemma shows that V is a convex increasing function in x. This property of the relative value function will play a crucial role in the derivation of the optimal policy. The next theorem shows how this lemma is used to obtain the structure of the optimal policy.

Theorem 3.1. *Assume $c_1(x)$ is a convex increasing function in x, $c_2(a)$ is an increasing function in a. Then the optimal resource allocation strategy is given by a non-decreasing curve, i.e., the optimal number of processors is given by $i^*(x)$ with $i^*(x)$ a non-decreasing function in x.*

Proof. Let $i^*(x)$ and $i^*(x+1)$ be the optimal allocation in states x and $x+1$, respectively. The proof is by contradiction. Suppose that $i^*(x+1) < i^*(x)$, then

$$\begin{aligned} T_{i^*(x+1)}(x) - T_{i^*(x)}(x) &= \big[\mu\big(i^*(x)\big) - \mu\big(i^*(x+1)\big)\big]\big[V(x) - V\big([x-1]^+\big)\big] - \big[c_2\big(i^*(x)\big) - c_2\big(i^*(x+1)\big)\big] \\ &\geq 0. \end{aligned}$$

By Lemma 3.2 we have that V is a convex, increasing function in x. Together with the fact that μ is an increasing function in a, and thus $\mu\big(i^*(x)\big) - \mu\big(i^*(x+1)\big) \geq 0$, we have

$$\begin{aligned} T_{i^*(x+1)}(x+1) - T_{i^*(x)}(x+1) &= \big[\mu\big(i^*(x)\big) - \mu\big(i^*(x+1)\big)\big]\big[V(x+1) - V(x)\big] - \big[c_2\big(i^*(x)\big) - c_2\big(i^*(x+1)\big)\big] \\ &\geq \big[\mu\big(i^*(x)\big) - \mu\big(i^*(x+1)\big)\big]\big[V(x) - V\big([x-1]^+\big)\big] - \big[c_2\big(i^*(x)\big) - c_2\big(i^*(x+1)\big)\big] \\ &\geq 0. \end{aligned}$$

However, this implies that $i^*(x+1)$ is not optimal for state $x+1$, since $i^*(x)$ has a a smaller value. Hence, $i^*(x+1) \geq i^*(x)$, and this establishes the non-decreasing curve as stated in the theorem. □

When additional assumptions are placed on the growth of processing costs c_2 and the service rate μ, one can specify the optimal policy in greater detail. The following theorem shows that if the processing costs are taken to be concave while the service rate is convex, then the optimal policy is a threshold-based bang–bang control policy.

Theorem 3.2. *Assume $c_1(x)$ is a convex increasing function in x, $c_2(a)$ is a concave increasing function in a, and $\mu(a)$ is a convex function in a. Then the optimal resource allocation strategy is a threshold-based bang–bang control policy, i.e., there exists a constant τ such that for $x < \tau$ the optimal action is to allocate no processors, and for $x \geq \tau$ the optimal action is to allocate all processors.*

Proof. Suppose that $c_1(x)$ is a convex increasing function in x and that $c_2(a)$ is a concave increasing function in a. To obtain the structure of the policy, consider Z_i for $i = 0, \ldots, A-1$ given by

$$Z_i(x) = T_i(x) - T_{i+1}(x) = \big[\mu(i+1) - \mu(i)\big]\big[V(x) - V\big([x-1]^+\big)\big] - \big[c_2(i+1) - c_2(i)\big].$$

Observe that for fixed x, the function $Z_i(x)$ is increasing in i since $\mu(i)$ is a convex function and $c_2(i)$ is concave. When $Z_i(x)$ is non-negative for some i, this means that using one processor more is better. However, since $Z_i(x)$ is increasing in i, this means that using all processors is optimal. Similarly, when $Z_{i-1}(x)$ is non-positive, then using one processor less is better. But since $Z_i(x)$ is an increasing function in i, using no processor at all is optimal. In order to establish the threshold τ, consider state $x+1$. If $Z_i(x)$ is non-negative, then $Z_i(x+1)$ is non-negative as well due to Lemma 3.2. Thus, when in state x it is optimal to use all processors, then in all states greater than x it is also optimal to use all processors. This establishes the bang–bang control policy, i.e., there exists a $\tau \geq 0$ such that the optimal policy for $x < \tau$ is to use no processors, and for $x \geq \tau$ using all processors is optimal. □

Theorem 3.2 shows that the optimal policy is either to use no processing capacity at all or to use all processors. The convexity/concavity properties of c_2 and μ are essential to obtain this result. When $c_2(a)$ is convex and $\mu(a)$ is concave in a, then this policy is not optimal anymore. In fact, the optimal policy is still of threshold type, but with more threshold levels (as specified by Theorem 3.1). However, the precise form heavily depends on the parameters used in the problem formulation. In the numerical experiments we will show that for some parameter values the policy behaves as a bang–bang policy, but for other parameters the multi-threshold control policy appears to be optimal.

3.2. Constrained Markov decision problem

In this section we describe the constrained Markov decision problem so that optimal policies can be determined that ensure that the total mean sojourn time of a customer in the whole system is less than α. Since the average time spent in the first step is β, the mean sojourn time in the second step is not allowed to exceed $\alpha' = \alpha - \beta$. In the previous subsection, we studied the unconstrained Markov decision problem in which we tried to find a policy π^* that minimized $g(\pi)$, i.e., the long-term average costs. We did not take into account the mean sojourn time constraint of a customer. Let W denote the sojourn time of an arbitrary customer in step 2. Then, the constrained Markov decision problem that we want to solve is

$$\min_{\pi} g(\pi) \quad \text{subject to } \mathbb{E}W \leq \alpha'.$$

Before the discussion of the general constrained case, we first examine the effect of the cost functions c_1 and c_2 on the optimal policy if one of them is set equal to zero.

No processing costs: When the processing costs are set to zero, i.e., $c_2(a) = 0$, then we obtain a situation that is equal to the case of the unconstrained Markov decision problem. The optimal policy is $a^* = A$, which (by assumption) clearly satisfies the service level α'.

No holding costs: When we assume $c_1(x) = 0$, then we cannot use the optimal policy of the unconstrained Markov decision problem: $a^* = 0$. In the case of the constrained problem, this would violate the service level constraint α'. For a given action a, the sojourn time in the system can be given by the sojourn time in an M/M/1 queue with arrival rate λ and service rate $\mu(a)$ (see [25]). Thus, let $\rho(a) = \lambda/\mu(a)$, then the expected sojourn time $\mathbb{E}W$ is given by

$$\mathbb{E}W = \frac{\rho(a)}{\lambda\big(1-\rho(a)\big)}.$$

The optimal action a^*, under the assumption of Poisson arrivals at station 2, is then obtained by solving the above equation for $\mathbb{E}W = \alpha'$. Note that in general this will result in a fractional value of a^*. This leads in a natural way to a randomized policy to obtain exactly α'. We will discuss the randomized policy further in the sequel.

We now proceed to the general solution of the constrained Markov decision problem. Note that the system basically resembles an M/M/1 queueing system with state-dependent service rates. For this system, due to Little's Law, we can relate the number of customers L in the system to the sojourn time in the system W by $\mathbb{E}L = \lambda\mathbb{E}W$ (see [26]). In order to get $\mathbb{E}L$, one can take x as cost function in the Markov decision problem. Hence, to obtain the expected sojourn time $\mathbb{E}W$ it suffices to have as cost function $c_3(x) = x/\lambda$. To solve the constrained problem we use the Lagrange multiplier approach described in Section 12.6 of Altman [27]. Let $\mathcal{L}$ be the Lagrange multiplier, then the dynamic programming operator T acting on $V_{\mathcal{L}}$ is defined as follows:

$$\begin{aligned} TV_{\mathcal{L}}(x) &= \lambda V_{\mathcal{L}}(x+1) + \min_{a\in\mathcal{A}}\big\{\mu(a)V_{\mathcal{L}}\big([x-1]^+\big) + \big(\mu(A)-\mu(a)\big)V_{\mathcal{L}}(x) + c_2(a)\big\} + c_1(x) + \mathcal{L}c_3(x) \\ &= \lambda V_{\mathcal{L}}(x+1) + \min_{a\in\mathcal{A}}\big\{\mu(a)V_{\mathcal{L}}\big([x-1]^+\big) + \big(\mu(A)-\mu(a)\big)V_{\mathcal{L}}(x) + c_2(a)\big\} + c_1^{\mathcal{L}}(x), \end{aligned} \tag{7}$$

with $c_1^{\mathcal{L}}(x) = c_1(x) + \mathcal{L}c_3(x)$ a convex increasing function. Note that the cost function is similar in structure to that of Eq. (5). Hence, Lemma 3.2, Theorems 3.1 and 3.2 apply to this case as well. Thus, for every value of $\mathcal{L}$ the optimal policy is a non-decreasing curve. Note that when for a given $\mathcal{L}$ an optimal policy has been derived by iterating Eq. (7), one can obtain the mean sojourn time $\mathbb{E}W$ by fixing the policy in Eq. (7) while setting $c_1 = c_2 = 0$ and $\mathcal{L} = 1$. Now, if the maximizing allocation actions in Eq. (7) increase with $\mathcal{L}$, and thus the rate to state $x-1$ increases while the rate to state x decreases in $TV_{\mathcal{L}}(x)$, then the value of $\mathbb{E}W$ decreases as well, since we have fewer customers on average in the system. This is demonstrated in the following lemmas.

Lemma 3.3. *$V_{\mathcal{L}}(x)$ is increasing in the Lagrange multiplier $\mathcal{L}$ for all $x \in \mathcal{X}$.*

Proof. To prove this lemma, we only need to prove that if $\mathcal{L}' > \mathcal{L}$ then $V_{\mathcal{L}'}(x) \geq V_{\mathcal{L}}(x)$ for all x. This is proven by induction in n. Let $V^0_{\mathcal{L}'}(x) = V^0_{\mathcal{L}}(x) = 0$ for all x. Note that $V^n_{\mathcal{L}}(x)$ is defined similarly to $V_n(x)$. Then, the lemma holds for $n = 0$. Assume it holds for $n = k$. We prove it also holds for $n = k+1$. Define $T^k_{a(\mathcal{L})}(x) = \mu(a)V_{\mathcal{L},k}([x-1]^+) + \big[\mu(A)-\mu(a)\big]V_{\mathcal{L},k}(x) + c_2(a)$. Then, for $x \geq 0$, we have

$$\begin{aligned} V_{\mathcal{L}',k+1}(x) - V_{\mathcal{L},k+1}(x) &= \lambda\big[V_{\mathcal{L}',k}(x+1) - V_{\mathcal{L},k}(x+1)\big] + \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x) + (\mathcal{L}'-\mathcal{L})c_3(x) \\ &\geq \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x). \end{aligned}$$

The inequality above holds because the term between the brackets is non-negative due to the induction hypothesis and the last term follows by the assumption $\mathcal{L}' > \mathcal{L}$. Let $a^* = \arg\min_a T^k_{a(\mathcal{L}')}(x)$. Then for $x \geq 0$, we have

$$\begin{aligned} V_{\mathcal{L}',k+1}(x) - V_{\mathcal{L},k+1}(x) &\geq T^k_{a^*(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x) \\ &\geq T^k_{a^*(\mathcal{L}')}(x) - T^k_{a^*(\mathcal{L})}(x) \\ &= \mu(a^*)V_{\mathcal{L}',k}([x-1]^+) + \big[\mu(A)-\mu(a^*)\big]V_{\mathcal{L}',k}(x) + c_2(a^*) \\ &\quad - \mu(a^*)V_{\mathcal{L},k}([x-1]^+) - \big[\mu(A)-\mu(a^*)\big]V_{\mathcal{L},k}(x) - c_2(a^*) \\ &= \mu(a^*)\big[V_{\mathcal{L}',k}([x-1]^+) - V_{\mathcal{L},k}([x-1]^+)\big] \\ &\quad + \big[\mu(A)-\mu(a^*)\big]\big[V_{\mathcal{L}',k}(x) - V_{\mathcal{L},k}(x)\big] \\ &\geq 0. \end{aligned}$$

Therefore, by induction we derive that $V_{\mathcal{L}'}(x) \geq V_{\mathcal{L}}(x)$. □

Lemma 3.3 shows that the relative value function is increasing in the Lagrange multiplier. This result is needed to show that the relative value function is also supermodular, as the following lemma shows.

Lemma 3.4. *If $\mathcal{L}' > \mathcal{L}$, then*

$$V_{\mathcal{L}'}(x+1) - V_{\mathcal{L}'}(x) - V_{\mathcal{L}}(x+1) + V_{\mathcal{L}}(x) > 0, \tag{8}$$

for all $x \in \mathcal{X}$.

Proof. This lemma is proven by induction. Let $V_{\mathcal{L}',0}(x) = \mathcal{L}'x$, and $V_{\mathcal{L},0}(x) = \mathcal{L}x$ for all x. Clearly, $V_{\mathcal{L}',0}(x+1) - V_{\mathcal{L}',0}(x) - V_{\mathcal{L},0}(x+1) + V_{\mathcal{L},0}(x) = \mathcal{L}' - \mathcal{L} > 0$. Assume it holds for $n = k$. Now, we prove it holds for $n = k+1$. Define $T^k_{a(\mathcal{L})}(x) = \mu(a)V_{\mathcal{L},k}([x-1]^+) + \big[\mu(A) - \mu(a)\big]V_{\mathcal{L},k}(x) + c_2(a)$. For all $x \geq 0$, we have

$$\begin{aligned}
&V_{\mathcal{L}',k+1}(x+1) - V_{\mathcal{L}',k+1}(x) - V_{\mathcal{L},k+1}(x+1) + V_{\mathcal{L},k+1}(x)\\
&\quad= \lambda\big[V_{\mathcal{L}',k}(x+2) - V_{\mathcal{L}',k}(x+1) - V_{\mathcal{L},k}(x+2) + V_{\mathcal{L},k}(x+1)\big]\\
&\qquad+ \min_a T^k_{a(\mathcal{L}')}(x+1) - \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x+1) + \min_a T^k_{a(\mathcal{L})}(x)\\
&\qquad+ \mathcal{L}'c_3(x+1) - \mathcal{L}'c_3(x) - \mathcal{L}c_3(x+1) + \mathcal{L}c_3(x)\\
&\quad> \min_a T^k_{a(\mathcal{L}')}(x+1) - \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x+1) + \min_a T^k_{a(\mathcal{L})}(x) + \big[\mathcal{L}' - \mathcal{L}\big]\big[c_3(x+1) - c_3(x)\big]\\
&\quad\geq \min_a T^k_{a(\mathcal{L}')}(x+1) - \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x+1) + \min_a T^k_{a(\mathcal{L})}(x).
\end{aligned}$$

The first inequality holds due to the induction hypothesis. The second inequality holds because $\mathcal{L}' > \mathcal{L}$ and $c_3(x)$ is an increasing function in x. Let $a^* = \arg\min_a T^k_{a(\mathcal{L}')}(x+1)$ and $b^* = \arg\min_a T^k_{a(\mathcal{L})}(x)$. Then we have

$$\begin{aligned}
&V_{\mathcal{L}',k+1}(x+1) - V_{\mathcal{L}',k+1}(x) - V_{\mathcal{L},k+1}(x+1) + V_{\mathcal{L},k+1}(x)\\
&\quad> T^k_{a^*(\mathcal{L}')}(x+1) - \min_a T^k_{a(\mathcal{L}')}(x) - \min_a T^k_{a(\mathcal{L})}(x+1) + T^k_{b^*(\mathcal{L})}(x)\\
&\quad\geq T^k_{a^*(\mathcal{L}')}(x+1) - T^k_{b^*(\mathcal{L}')}(x) - T^k_{a^*(\mathcal{L})}(x+1) + T^k_{b^*(\mathcal{L})}(x)\\
&\quad= \mu(a^*)\big[V_{\mathcal{L}',k}(x) - V_{\mathcal{L},k}(x)\big] + \big[\mu(A) - \mu(a^*)\big]\big[V_{\mathcal{L}',k}(x+1) - V_{\mathcal{L},k}(x+1)\big]\\
&\qquad- \mu(b^*)\big[V_{\mathcal{L}',k}(x-1) - V_{\mathcal{L},k}(x-1)\big] - \big[\mu(A) - \mu(b^*)\big]\big[V_{\mathcal{L}',k}(x) - V_{\mathcal{L},k}(x)\big]\\
&\quad= \mu(a^*)\big[V_{\mathcal{L}',k}(x) - V_{\mathcal{L},k}(x) - V_{\mathcal{L}',k}(x+1) + V_{\mathcal{L},k}(x+1)\big]\\
&\qquad+ \mu(b^*)\big[V_{\mathcal{L}',k}(x) - V_{\mathcal{L},k}(x) - V_{\mathcal{L}',k}(x-1) + V_{\mathcal{L},k}(x-1)\big]\\
&\qquad+ \mu(A)\big[V_{\mathcal{L}',k}(x+1) - V_{\mathcal{L},k}(x+1) - V_{\mathcal{L}',k}(x) + V_{\mathcal{L},k}(x)\big]\\
&\quad> \big[\mu(A) - \mu(a)\big]\big[V_{\mathcal{L}',k}(x+1) - V_{\mathcal{L},k}(x+1) - V_{\mathcal{L}',k}(x) + V_{\mathcal{L},k}(x)\big] \geq 0.
\end{aligned}$$

Hence, using mathematical induction we have proved the lemma. □

We are now ready to prove the monotonicity result of the relative value function, i.e., when the Lagrange multiplier increases, and thus places more emphasis on the holding costs, the number of allocated processors also increases. The following theorem formalizes this statement.

Theorem 3.3. *Let $T_{a(\mathcal{L})}(x) = \mu(a)V_{\mathcal{L}}([x-1]^+) + \big[\mu(A) - \mu(a)\big]V_{\mathcal{L}}(x) + c_2(a)$. If $\mathcal{L}' > \mathcal{L}$, then $\arg\min_a T_{a(\mathcal{L}')}(x) \geq \arg\min_a T_{a(\mathcal{L})}(x)$ for all $x \in \mathcal{X}$.*

Proof. The proof is by contradiction. Suppose that $a' := \arg\min_a T_{a(\mathcal{L}')}(x) < a := \arg\min_a T_{a(\mathcal{L})}(x)$. Then, since $T_{a(\mathcal{L}')}(x) \geq T_{a'(\mathcal{L}')}(x)$, we have

$$\begin{aligned}
T_{a(\mathcal{L}')}(x) - T_{a'(\mathcal{L}')}(x) &= \big[\mu(a') - \mu(a)\big]\big[V_{\mathcal{L}'}(x) - V_{\mathcal{L}'}\big([x-1]^+\big)\big] - \big[c_2(a') - c_2(a)\big]\\
&\geq 0.
\end{aligned}$$

Together with relation (8) and using the fact that μ is an increasing function, we have

$$\begin{aligned}
T_{a(\mathcal{L})}(x) - T_{a'(\mathcal{L})}(x) &= \big[\mu(a') - \mu(a)\big]\big[V_{\mathcal{L}}(x) - V_{\mathcal{L}}\big([x-1]^+\big)\big] - \big[c_2(a') - c_2(a)\big]\\
&> \big[\mu(a') - \mu(a)\big]\big[V_{\mathcal{L}'}(x) - V_{\mathcal{L}'}\big([x-1]^+\big)\big] - \big[c_2(a') - c_2(a)\big]\\
&\geq 0,
\end{aligned}$$

which implies that $a \neq \arg\min_a T_{a(\mathcal{L})}(x)$, which is in contradiction with the assumption. □

There is a difference in the optimal policy between the one in the unconstrained case and in the constrained case. Note that due to Theorem 3.3 a higher value of the Lagrange multiplier $\mathcal{L}$ implies that the mean sojourn time $\mathbb{E}W$ decreases because more processors are allocated. Therefore, there is a $\mathcal{L}^*$ for which $\mathbb{E}W_{\mathcal{L}^*} \geq \alpha'$ and $\mathbb{E}W_{\mathcal{L}^*+\epsilon} < \alpha'$ for a small $\epsilon > 0$, where $\mathbb{E}W_{\mathcal{L}}$ is the mean sojourn time under Lagrange parameter $\mathcal{L}$. Observe that $\mathbb{E}W_{\mathcal{L}}$ is not a continuous function of $\mathcal{L}$, because the optimal policy does not change continuously with $\mathcal{L}$ but changes at specific values. In the constrained case, the optimal policy is to randomize between the associated policies $\pi(\mathcal{L}^*)$ and $\pi(\mathcal{L}^* + \epsilon)$ so that exactly $\mathbb{E}W = \alpha'$ is achieved. The optimal policy thus randomizes between two step-function policies (see [27] for the proof).

Note that we analyzed the optimal actions for station 1 independently of the state at station 2. However, in our formulation we can see that the first station has influence on the policy in station 2. The higher the time β is spent at station 1, the tighter the constraint in station 2 becomes through $\alpha' = \alpha - \beta$. In the next section we shall illustrate how the optimal policy for the constrained problem is derived and how this tighter constraint affects the optimal policy.

4. Numerical experiments

In the previous sections we dealt with station 1 and station 2 separately. Although, the analysis has been done separately, the performance of the system of both stations, in reality, is dependent on each other as was shown in Section 3.2. In this section we will validate the performance of the decomposed model in which we split up the real system with the simulated performance of the real system and perform a numerical comparison of the optimality. We will illustrate how variability in the service duration R and the set-up time T affects the policy in station 2.

4.1. Accuracy of the approximation

In Section 2 we described the model as a system in which customers get service in two steps. First, the customer receives service in step 1 in which a controller has to decide on the reservation time. Then, the customer, while still blocking the service facilities in step 1, receives service in step 2 in which a controller has to decide on the number of allocated computing resources. We have split up this system into two parts and added the sojourn times of each part to each other. In doing so we make an error in the performance as compared to the real system as described in the end of Section 2.

First, we study how big the error is in the performance when we split up the system. To this end, let w_{real} denote the mean sojourn time of the real system (that we obtain by simulation), and w_{model} denote the mean sojourn time that a customer spends in steps 1 and 2 together as obtained from the decomposed model. Note that the simulations have been run sufficiently long to ensure that the results shown in the following tables are significant. Table 1 depicts the relative error

$$\Delta_w := \frac{w_{\text{real}} - w_{\text{model}}}{w_{\text{real}}} \times 100\%,$$

when we split up the system, for various values of λ, μ, δ, and γ (assuming exponential distributions for R, T) using the policy that achieves $w_{\text{model}} = \alpha$ under minimal costs obtained in the decomposed model, with the cost functions specified in experiment 1 of Table 3. The results of Table 1 show that the approximation works particularly well for high values of (δ, γ), but tends to degrade for smaller values. This is due to the fact that the two-step approach neglects the potential waiting time for availability of the service in the first step upon arrival epochs. This effect becomes more apparent for smaller values of (δ, γ) and for a higher utilization of the system. Note that the higher values of (δ, γ) (i.e., the service times in step 1 are relatively short in comparison) reflect realistic values of system parameters that are used in real applications (e.g., iris scan and fingerprint applications). This underlines the applicability of the decomposed model in practice.

Second, we assess the accuracy of the approximation for our cost function defined in (1). To this end, let c_{real} denote the mean cost of the real system (that we obtain by simulation), and c_{model} denote the mean cost that a customer encounters in steps 1 and 2 together as obtained from the decomposed model. Table 2 depicts the relative error

$$\Delta_c := \frac{c_{\text{real}} - c_{\text{model}}}{c_{\text{real}}} \times 100\%,$$

under the same parameter scenarios and policies as used in Table 1, and with the cost functions defined as in Table 3. Table 2 shows similar results as those in Table 1, namely the approximation works well for high values of (δ, γ) and degrades for smaller values. The different signs in the table indicate that the costs curves for the real system and the model do not completely coincide, but are slightly shifted relative to each other. The insight obtained from Tables 1 and 2 is that for our problem (1), both the cost and the sojourn time show similar behavior, in particular for high values of (δ, γ). This insight suggests that the optimal policy in the decomposed model is a good approximation to the optimal policy in the real system. In order to validate this, we compare the performance under the optimal policy in the real system with the performance in the decomposed model. The derivation of the optimal policy in the real system is computationally intractable for large values of A. Therefore, we construct a small-sized problem with $A = 5$ such that enumerating over *all relevant* values of s and *all* monotone increasing step functions for the processor allocation becomes tractable.

The results of this experiment are in agreement with the results of Tables 1 and 2. To explain in greater detail where differences in the performance occur, we discuss the extreme scenarios in more detail below. We focus on $\lambda = 0.5$, $\mu = 1.2$, and the two scenarios $(\delta, \gamma) = (8, 12)$ and $(\delta, \gamma) = (44, 64)$. In the case with $(\delta, \gamma) = (8, 12)$, the unconstrained

Table 1
The relative error Δ_w for different parameter values with $\mu(a) = \mu a$.

$\lambda = 0.5$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	13.0	8.5	6.3	4.9	4.1	3.5	2.9	2.5	2.1	2.1
1.2	7.6	4.8	3.6	2.7	2.3	1.9	1.6	1.4	1.2	1.1
2.0	8.6	5.7	4.2	3.2	2.6	2.3	2.0	1.7	1.5	1.4
4.0	8.2	5.5	4.4	3.5	2.9	2.6	2.3	2.0	1.9	1.7
$\lambda = 0.7$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	22.8	15.1	11.3	8.8	7.2	6.3	5.3	4.9	4.3	4.1
1.2	12.0	7.5	5.6	4.3	3.5	2.9	2.5	2.2	2.0	2.0
2.0	10.5	6.5	4.9	3.7	3.0	2.5	2.2	1.9	1.7	1.6
4.0	11.6	7.8	6.0	4.8	4.1	3.6	3.2	2.8	2.5	2.4
$\lambda = 1.0$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	38.0	25.9	19.4	15.5	12.6	10.9	9.5	8.5	7.6	7.1
1.2	20.0	12.7	9.4	7.5	6.0	5.0	4.4	3.8	3.5	3.2
2.0	17.8	11.4	8.3	6.5	5.3	4.5	3.9	3.4	3.1	2.8
4.0	17.1	11.3	8.6	6.8	5.7	4.7	4.1	3.6	3.2	2.9

Table 2
The relative error Δ_c for different parameter values with $\mu(a) = \mu a$.

$\lambda = 0.5$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	10.24	6.44	4.36	3.18	2.81	2.20	1.69	1.23	1.16	1.20
1.2	3.21	1.45	0.79	0.35	0.23	0.09	0.01	0.21	0.20	−0.03
2.0	1.89	0.17	−0.18	−0.23	−0.51	−0.26	−0.38	−0.36	−0.41	−0.20
4.0	3.79	1.24	0.30	0.12	−0.23	−0.21	−0.27	−0.33	−0.24	−0.39
$\lambda = 0.7$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	19.23	11.98	8.58	6.78	5.34	4.48	3.99	3.61	3.15	2.75
1.2	7.66	3.77	2.36	1.70	1.48	0.80	0.79	0.69	0.63	0.63
2.0	5.73	2.25	1.25	0.58	0.12	0.29	0.21	0.25	0.04	0.06
4.0	6.11	2.00	0.77	0.26	0.01	−0.22	−0.26	−0.19	−0.28	−0.32
$\lambda = 1.0$	(δ, γ)									
μ	(8, 12) (%)	(12, 20) (%)	(16, 24) (%)	(20, 32) (%)	(24, 40) (%)	(28, 48) (%)	(32, 56) (%)	(36, 64) (%)	(40, 64) (%)	(44, 64) (%)
0.4	35.41	22.69	16.37	13.38	10.36	8.58	7.57	6.58	6.00	5.02
1.2	17.26	9.15	6.08	4.63	3.66	2.95	2.56	1.93	1.71	1.77
2.0	13.07	6.14	3.64	2.15	1.71	1.26	1.12	1.01	0.79	0.59
4.0	11.81	4.32	2.01	1.05	0.22	−0.04	−0.15	−0.33	−0.43	−0.25

Table 3
Functions for c_1, c_2, and μ.

Experiment 1	$c_1(x) = 10x^2$	$c_2(a) = 10a^2$	$\mu(a) = \sqrt{a}\mu$
Experiment 2	$c_1(x) = \frac{1}{100}x^2$	$c_2(a) = 100\sqrt{a}$	$\mu(a) = a^2\mu$

optimization leads to a relative difference of 2.6% in cost (Δ_c) and 7.7% in mean sojourn time (Δ_w). When we add a constraint with $\alpha = 1.2$, the differences are 8.0% and 7.7% for the cost and mean sojourn time, respectively. In the case with $(\delta, \gamma) = (44, 64)$, the unconstrained optimization has relative differences 0.58% and 1.85% for the cost and sojourn time, respectively. For the constrained problem with $\alpha = 1.06$, the relative differences are 0.02% and 0.93%. In conclusion, the approximate two-step approach has a reasonable performance, and works particularly for realistic parameter values.

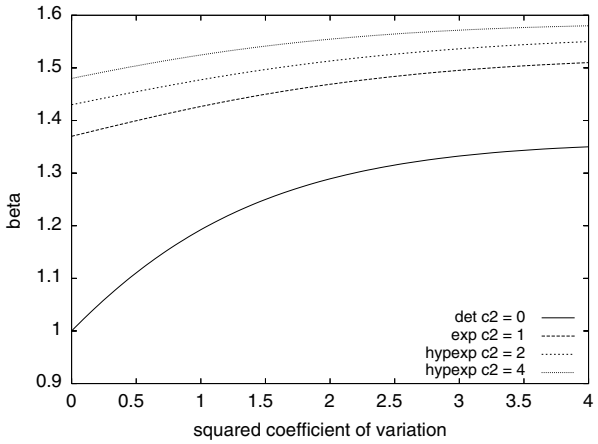
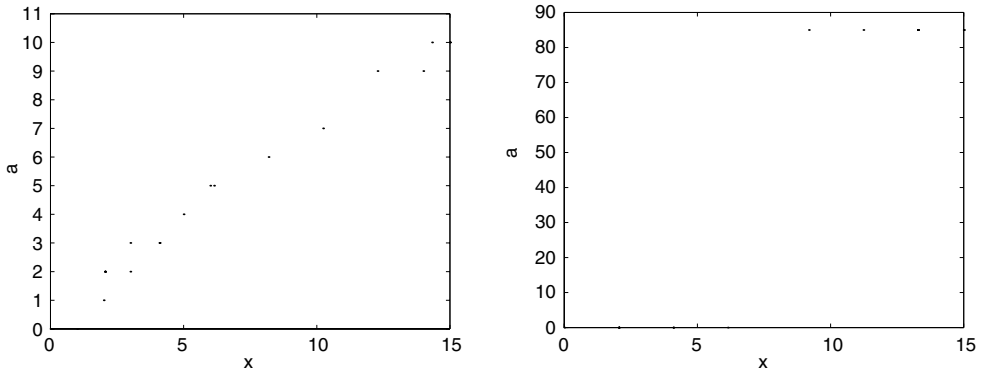


Fig. 4. The time spent at station 1 for different distributions of R and T.



(a) Illustration of Theorem 3.1. (b) Illustration of Theorem 3.2.

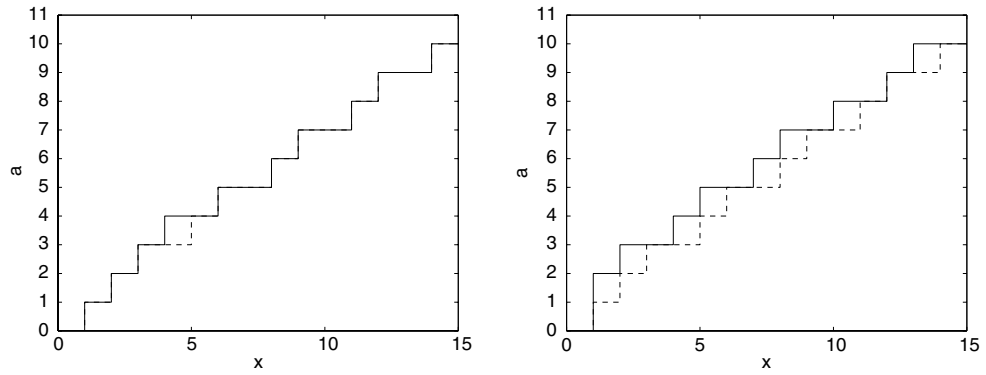
Fig. 5. Numerical experiments to illustrate the structure of the optimal policy.

4.2. The effect of variability in R and T

In this section, we illustrate the effect of the variability in R and T on the time spent at station 1. By Lemma 3.1 we know that s^* only depends on these variables through their mean, but the time spent at the station also takes into consideration the second moment of R and T. Therefore, we take for R and T distributions that increase in the coefficient of variation c^2. More specifically, we take for R and T a deterministic, exponential, hyper-exponential, and hyper-exponential distribution with $c^2 = 0$, 1, 2, and 4, respectively, while keeping the mean constant. Fig. 4 shows β, obtained by explicitly calculating $\mathbb{E}\max\{R, s^* + T\}$, as a function of the c^2 for T (with mean $2/3$). The four curves correspond to the different distributions of R (with fixed mean 1). We also did experiments with distributions for T with means 1 and $6/5$, and they gave similar curves. The graph shows that the increase in the coefficient of variation for R and/or T results in a larger sojourn time β that grows in a non-linear concave manner.

Now that we know how the distributions affect the β and thus the constraint level $\alpha' = \alpha - \beta$, we focus our attention to station 2 to gain further insight into the optimal resource allocation policy in the decomposed problem. Consider the following parameters for this station: $\lambda = 0.5$, $\mu = 0.7$, and $A = 85$ (this represents the number of processors in the DAS-3 cluster that we have used, see [23]). To illustrate the structure of the policies, we vary the structure of the functions for c_1, c_2, and μ as given in Table 3. Experiments 1 and 2 satisfy the conditions of Theorems 3.1 and 3.2, respectively. The corresponding optimal resource policies in the decomposed problem, that are obtained through value iteration as described in Section 3.1, are illustrated in Fig. 5(a) and (b). Fig. 5(a) shows that the optimal resource strategy a^* is a non-decreasing function in x, while Fig. 5(b) illustrates the bang–bang control policy.

Fig. 5 illustrates the optimal policy in case of the unconstrained Markov decision problem, and the service level constraint is thus not taken into account. When the service level constraint is added, the policy changes to a more conservative strategy in which more processors are used to provide the guarantee on the sojourn time. To get insight into how the optimal policy changes, we vary $\mathcal{L}$ to study this effect for experiment 1 in Table 3. For $\mathcal{L} = 0$ we get the unconstrained policy that is already depicted in Fig. 5(a) with $g = 43.655$ and $\mathbb{E}W = 1.987$. As $\mathcal{L}$ increases, the policy changes at $\mathcal{L} = 0.94$ with corresponding values $g = 43.690$ and $\mathbb{E}W = 1.949$. Fig. 6(a) shows how the policy differs from the previous policy; the dotted line is the



(a) Results for $\mathcal{L} = 0.94$.

(b) Results for $\mathcal{L} = 5.00$.

Fig. 6. Numerical experiments to illustrate the effect of changing values of $\mathcal{L}$.

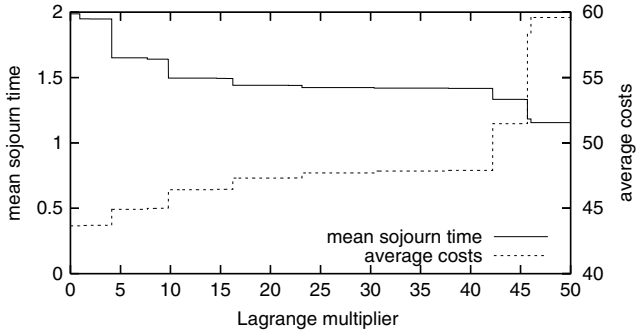


Fig. 7. The average cost and the sojourn time as a function of the Lagrange multiplier.

policy for $\mathcal{L} = 0$ and the solid line depicts the policy for $\mathcal{L} = 0.94$. The policy differs in state 4 by using an additional server to serve the customers. Hence, to achieve $1.949 < \mathbb{E}W < 1.987$, one has to randomize between the two policies. As $\mathcal{L}$ increases, the policy becomes more conservative, in the sense that more computing resources are used in an earlier stage. Fig. 6(b) depicts exactly this situation where $\mathcal{L} = 5.00$ is used.

To better understand the trade-off in the reduction in the average costs g and the improvement in the mean sojourn time, we depict both quantities as a function of $\mathcal{L}$ in Fig. 7. The figure shows that in the initial increment of $\mathcal{L}$ the improvement in the mean sojourn time is the biggest, whereas the biggest increase in the average costs occurs for even bigger values of $\mathcal{L}$. Hence, one can balance both quantities by choosing an appropriate value of $\mathcal{L}$ such that the biggest improvements in the mean sojourn time are obtained while the increment in costs is relatively small. Moreover, the figure can also be used to determine the right value of $\mathcal{L}$ for a given α'. In combination with Fig. 4 this provides a complete picture of how the variability in R and T affects the approximation with respect to the allocation of processors in station 2.

5. Conclusion and further discussion

In this paper we have explored the optimal resource allocation problem in time-reservation systems. In such systems one needs to optimize the resource-allocation costs and reservation moments simultaneously on one hand while satisfying a QoS constraint on the sojourn time of a job on the other hand. We decompose the problem into two parts and derive an optimal policy. This optimal policy serves as an approximation for the optimal policy in the original problem posed. For the decomposed problem, we first showed that the optimal reservation moment is given by the difference of the mean service time and the mean reservation time. However, the sojourn time at the first station does take into account the second moment as well. Then, we applied dynamic programming to show that the optimal resource allocation policy in the decomposed problem follows a step function with as extreme policy the bang–bang control for given structures of the cost function and the service rate. Extensive numerical experiments show that the optimal policy in the decomposed problem has nearly optimal performance as compared to the performance of the optimal policy in the real system.

Next, we mention several interesting avenues for further research. The work described in this paper is part of a much larger realistic problem that exists in practice. In our current work, we focus on only one facility that uses a number of computing resources to provide a service under a QoS-constraint to its customers, whereas in practice more than one facility

may exist sharing the same computing resources. This creates dependence between the different facilities and the decision maker may not observe the state of all facilities. This warrants new stochastic control methodologies that deal with this dependence and partial information in which the insights obtained in this paper can be very useful. Moreover, the decisions can be taken in a centralized or a decentralized manner.

In addition to having multiple facilities, the number of computing nodes may not be sufficient in all situations. In view of the limited processing capacity new interesting problems from the server's point of view need to be addressed. (1) one could dynamically add computing nodes to the system. However, this takes a significant setup time which needs to be taken into account. When does the system need to add/remove processing capacity? (2) in the light of time-varying parameters, such as the arrival rates, the algorithm also needs to be robust avoiding oscillations through, e.g., hysteresis. (3) The limited processing capacity also comes with a scheduling problem when many facilities share the same resource.

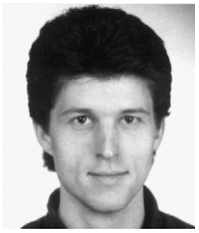
References

[1] Grosu Daniel, Anthony T. Chronopoulos, Algorithmic mechanism design for load balancing in distributed systems, IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics) (2003) 1–8.
[2] Jonghun Park, A scalable protocol for deadlock and livelock free co-allocation of resources in internet computing, in: Proceeding of the Symposium on Applications and the Internet, 2003, pp. 66–73.
[3] Omer F. Rana, Michael Winikoff, Lin Padgham, James Harland, Applying conflict management strategies in BDI agents for resource management in computational grids author, in: Proceedings of the 8th International Workshop on Quality of Service (IWQOS), IEEE Computer Society Press, 2002, pp. 205–214.
[4] Karl Czajkowski, Ian T. Foster, Carl Kesselman, Resource co-allocation in computational grids, in: Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing, HPDC-8, 1999, pp. 219–228.
[5] The globus alliance. http://www.globus.org.
[6] I. Foster, A. Roy, V. Sander, A quality of service architecture that combines resource reservation and application adaptation, in: Proceedings of the 8th International Workshop on Quality of Service, IWQOS, Pittsburgh, PA, June 2000, pp. 181–188.
[7] A. Roy, V. Sander, Grid resource management: state of the art and future trends, in: GARA: A Uniform Quality of Service Architecture, Kluwer Academic Publishers, 2004, pp. 373–394.
[8] Xiaozhi Wang, Junzhou Luo, Architecture of grid resource allocation management based on QoS, Grid and Cooperative Computing (2004) 81–88.
[9] Thomas Sandholm, Kevin Lai, Jorge Andrade, Jacob Odeberg, Market-based resource allocation using price prediction in a high performance computing grid for scientific applications, in: Proceedings of the IEEE International Symposium on High Performance Distributed Computing, HPDC, Paris, France, 2006, pp. 19–23.
[10] M. Feldman, K. Lai, L. Zhang, A price-anticipating resource allocation mechanism for distributed shared clusters, in: Proceedings of the ACM Conference on Electronic Commerce, 2005.
[11] Rajkumar Buyya, David Abramson, Jonathan Giddy, A case for economy grid architecture for service oriented grid computing, in: Proceedings on the 15th International Symposium on Parallel and Distributed Processing, April 2001, pp. 776–790.
[12] Brent Chun, David Culler, Market-based proportional resource sharing for clusters, Technical Report, University of California, Berkeley, September 1999.
[13] R. Cocchi, S. Shanker, D. Estrin, L. Zhang, Pricing in computer networks: motivation, formulation, and example, IEEE/ACM Transactions on Networking 1 (6) (1993) 614–627.
[14] A. Lazar, N. Semret, Auctions for network resource sharing, Technical Report TR 468-97-02, Columbia University, February 1997.
[15] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, C. Staelin, An economic paradigm for query processing and data migration in mariposa, in: Proceedings of 3rd International Conference on Parallel and Distributed Information Systems, IEEE Comput. Soc. Press, Austin, TX, USA, 1994, pp. 28–30.
[16] Abdul Aziz, Hesham El-Rewini, Grid resource allocation and task scheduling for resource intensive applications, in: Proceedings of the 2006 International Conference on Parallel Processing Workshops, ICPPW'06, 2006.
[17] D. Nurmi, R. Wolski, J. Brevik, Probabilistic advanced reservations for batch-scheduled parallel machines, in: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP 2008, Salt Lake City, UT, USA, vol. 20–23, February 2008, pp. 289–290.
[18] D. Nurmi, J. Brevik, R. Wolski, QBETS: queue bounds estimation from time series, in: Proceedings of Job Scheduling Strategies for Parallel Processing, JSSPP, June 2007.
[19] K. Fukuda, N. Wakamiya, M. Murata, H. Miyahara, Integrated resource allocation for real-time video transfer to maximize user's utility, IEEE Journal on Selected Areas in Communications (1999).
[20] D. Nurmi, R. Wolski, J. Brevik, VARQ: virtual advance reservations for queues, in: Proceedings of the 17th International Symposium on High Performance Distributed Computing, 2008, pp. 75–86.
[21] S. Stidham Jr., R.R. Weber, A survey of Markov decision models for control of networks of queues, Queueing Systems 13 (1993) 291–314.
[22] A. Mutz, R. Wolski, J. Brevik, Eliciting honest value information in a batch-queue environment, in: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, IEEE Computer Society, 2007, pp. 291–297.
[23] The distributed ASCI supercomputer 3. http://www.cs.vu.nl/das3.
[24] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 1994.
[25] R.B. Cooper, Introduction to Queueing Theory, North Holland, 1981.
[26] J.D.C. Little, A proof of the queueing formula $L = \lambda W$, Operations Research 9 (1961) 383–387.
[27] E. Altman, Constrained Markov Decision Processes, Chapman and Hall, 1999.

Ran Yang received her Master of Science degree from VU University Amsterdam, the Netherlands in 2004. Then she joined industry for 1 year. Since 2006, she is a Ph.D. researcher in Faculty of Sciences in VU University Amsterdam, The Netherlands and PNA2 (Probability and Stochastic Networks) department in CWI (Centre of Mathematics and Computer Science), Amsterdam, The Netherlands.

Sandjai Bhulai (1976) received his M.Sc. degrees in Mathematics and in Business Mathematics and Informatics, both from the VU University Amsterdam, The Netherlands. He carried out his Ph.D. research on "Markov decision processes: the control of high-dimensional systems" at the same university for which he received his Ph.D. degree in 2002. After that he has been a postdoctoral researcher at Lucent Technologies, Bell Laboratories as NWO Talent Stipend fellow. In 2003 he joined the Mathematics department at the VU University Amsterdam, where he is an associate professor in Applied Probability and Operations Research. His primary research interests are in the general area of stochastic modeling and optimization, in particular, the theory and applications of Markov decision processes. His favorite application areas include telecommunication networks and call centers. He is currently involved in the control of time-varying systems, partial information models, dynamic programming value functions, and reinforcement learning.

Rob van der Mei is the leader of the research cluster Probability, Networks and Algorithms (PNA within CWI, and a (part-time) full professor at the VU University, Amsterdam. Before going to academia, he has been working for over a decade as a consultant and researcher in the ICT industry, working for PTT, KPN, AT&T Bell Labs and TNO ICT. He has been a member of the editorial board of Performance Evaluation and the AEUE Journal on Electronics and Communications. He is a co-founder and general chair of the recently recognized ICT Innovation Platform (IIP) "Vital ICT Infrastructures", a platform for exchanging knowledge and experience in the area of QoS of ICT systems between academia a wide range of ICT companies. He is a co-founder and board member of the recently recognized Dutch Mathematics cluster Stochastics – Theoretical and Applied Research (STAR). His research interests include performance modeling and scalability analysis of ICT systems, logistics, grid computing, revenue management, sensor networks and queueing theory. He is the (co-)author of over 90 papers in journals and refereed proceedings.



Frank Seinstra is an assistant professor in the Department of Computer Science at VU University Amsterdam, working in the High Performance Distributed Systems research group headed by Prof. H.E. Bal. In 2003, he received his Ph.D. degree from the University of Amsterdam for his research on "User Transparent Parallel Image Processing". His current research efforts are aimed at the design of programming models for large-scale heterogeneous parallel and distributed systems, with a specific focus on the application and research domain of multimedia content analysis.