Optimal Admission Control for a QoS-aware Service-oriented System

Marco Abundo, Valeria Cardellini, and Francesco Lo Presti

DISP, Università di Roma "Tor Vergata", Italy
marco.abundo@gmail.com, cardellini@ing.uniroma2.it,
lopresti@info.uniroma2.it

Abstract. In the service computing paradigm, a service broker can build new applications by composing network-accessible services offered by loosely coupled independent providers. In this paper, we address the admission control problem for a a service broker which offers to prospective users a composite service with a range of different Quality of Service (QoS) classes. We formulate the problem as a Markov Decision Process (MDP) problem with the goal of maximizing the broker revenue while guaranteeing non-functional QoS requirements to its already admitted users. To assess the effectiveness of the MDP-based admission control, we present experimental results where we compare the optimal decisions obtained by the analytical solution of the MDP with other policies.

1 Introduction

In the Service Oriented Architecture (SOA) paradigm, the design of complex software is facilitated by the possibility to build new applications by composing loosely-coupled services. The so built composite service is offered by a *service broker* to different classes of users characterized by diverse Quality of Service (QoS) requirements. To this end, the broker and its users generally engage in a negotiation process, which culminates in the definition of a *Service Level Agreement* (SLA) about their respective duties and QoS expectations.

In the upcoming Internet service marketplace, multiple service providers may offer similar competing services corresponding to a functional description but at differentiated levels of QoS and cost. Therefore, in undertaking the management of the SOA-based system that offers the composite service, the broker has to select at runtime the best set of component services implementing the needed functionalities (with which we assume the service broker has contracted a SLA specifying the QoS the component service will provide to the broker) in order to maximize some utility goal (e.g., its revenue) while guaranteeing the QoS levels to the composite service users.

A significant number of research efforts have been devoted to service selection issues, e.g., [1, 6, 7]. The common aim of these works is to identify for each

abstract functionality in the composite service a pool (eventually a singleton) of corresponding concrete services, selecting them from a set of candidates. All these efforts implicitily assume that a user is admitted to the service as long as enough resources are available to serve its requests at the required QoS level without violating existing users' QoS. Nevertheless, it is not surprising that such a simple policy, which we will call hereafter *blind*, might not turn to be optimal with respect to maximizing the broker utility.

In this paper, to overcome the limitations of the aforementioned results we study the admission problem for the *MOdel-based SElf-adaptation of SOA systems* (MOSES) service broker we proposed in [4,7], which manages a composite service offering differentiated QoS service classes. We formulate the optimal admission control problem as a Markov Decision Process (MDP) problem with the goal to maximise the broker discounted expected infinite horizon reward while guaranteeing non-functional QoS requirements to its users. Our results show the MDP-based admission control always guarantees significantly higher average rewards than the *blind* policy. We also considered finite horizon policies which are computationally more efficient than the infinite horizon counterpart and that allow us to tradeoff optimality with computational complexity/time-horizon length. Our findings show that even with the simple 1-step horizon policy it is possible to achieve better results with respect to the *blind* policy, quite close to the infinite horizon optimum, but at a fraction of the computational cost.

A considerable number of research efforts have focused on the application of MDP-based models and stochastic programming to SOA systems and, more generally, to software systems [2,3,5,8–10,12,14]. Some of these works have proposed self-healing approaches in order to support the reconfiguration of running services, e.g., [3,8], considering also proactive solutions for SOA systems [12]. Some recent approaches [9,10,14] have used MDPs to model service composition with the aim to create automatically an abstract workflow of the service composition that satisfies functional and non-functional requirements, and also to allow the composite service to adapt dynamically to a varying environment [14]. Some of the aforementioned works have proposed MDP-based admission control in service-oriented systems [2,5] and are therefore most closely related to ours. In [2] Bannazadeh and Leon-Garcia have proposed an admission control for service-oriented systems which uses an online optimization approach for maximizing the system revenue, while in [5] Bichler and Setzer have applied an MDP-based formulation to tackle admission control for media on demand services. However, both these works do not consider composite services organized according to some business logic, while our approach is able to manage the admission control for a composite service whose workflow entails the composition patterns typical of orchestration languages such as BPEL [11], which is the de-facto standard. To the best of our knowledge, the approach we propose in this paper is the first admission control policy based on MDPs for QoS-aware composite services.

The rest of the paper is organized as follows. In Section 2 we present the SOA system managed by the service broker. In Section 3 we present our MDP problem. Then, in Section 4 we sketch out the implementation of our admission

policies and present the simulation experiments to assess the effectiveness of the proposed MDP-based approach. Section 5 concludes the paper.

2 MOSES System

MOSES is a QoS-driven runtime adaptation framework for SOA-based systems, designed as a service broker. In this section, we provide an overview on the MOSES system for which we propose in this paper MDP-based admission control policies. A detailed description of the MOSES methodology, architecture and implementing prototype can be found in [7] and [4], respectively.
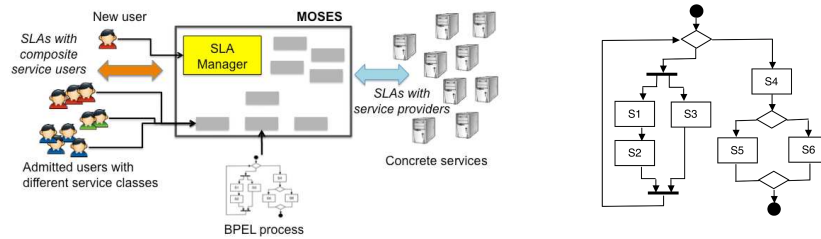

Fig. 1. MOSES operating eveninronment (left); MOSES-compliant workflow (right).

MOSES acts as a third-party intermediary between service users and providers, performing a role of provider towards the users and being in turn a requestor to the providers of the concrete services. It advertises and offers the composite service with a range of service classes which imply different QoS levels and monetary prices. Figure 1 (left) shows a high-level view of the MOSES environment, where we have highlighted the MOSES component on which we focus in this paper, i.e., the *SLA Manager*. The workflow that defines the composition logic of the service managed by MOSES can include all the different types of BPEL structured activities: `sequence`, `switch`, `while`, `pick`, and `flow` [11]. Figure 1 (right) shows an example of BPEL workflow, described as a UML2 activity diagram, that can be managed by MOSES. The figure also shows the functionalities (named *tasks* and represented by $S_1, \ldots, S_6$) needed to compose the new added value service.

MOSES performs a two-fold role of service provider towards its users, and of service user with respect to the providers of the concrete services it uses to implement the composite service it is managing. Hence, it is involved in two types of SLAs, corresponding to these two roles. MOSES presently considers the average value of the following attributes:

- *response time*: the interval of time elapsed from the service invocation to its completion;
- *reliability*: the probability that the service completes its task when invoked;

– *cost*: the price charged for the service invocation.

Our general model for the SLA between the provider and the user of a service consists of a tuple $\langle T, C, R, L \rangle$, where: T is the upper bound on the average service response time, C is the service cost per invocation, R is the lower bound on the service reliability. The provider guarantees that thresholds T and R will hold on average provided that the user request rate does not exceed L.

In the case of the SLAs between the composite service users and MOSES (acting the provider role), we assume that MOSES offers a set K of service classes. Hence, the SLA for each user u of a class $k \in K$ is defined as a tuple $\langle T_{\max}^k, C^k, R_{\min}^k, L_u^k, P_\tau^k, P_\rho^k \rangle$. The two additional parameters P_τ^k and P_ρ^k represent the penalty rates MOSES will refund its users with for possible violations of the service class response time and reliability, respectively. All these coexisting SLAs (for each u and k) define the QoS objectives that MOSES must meet. We observe that MOSES considers SLAs stating conditions that should hold globally for a *flow* of requests generated by a user. To meet these objectives, we assume that MOSES (acting the user role) has already identified for each task $S_i \in \mathcal{F}$ in the composite service a pool of corresponding concrete services implementing it. The SLA contracted between MOSES and the provider of the concrete service $i.j \in I_i$ is defined as a tuple $\langle t_{ij}, c_{ij}, r_{ij}, l_{ij} \rangle$. These SLAs define the constraints within which MOSES should try to meet its QoS objectives.

New users requesting the composite service managed by MOSES are subject to an accept/deny decision, with which MOSES determines whether or not it is convenient to admit the user in the system according to the user SLA and the system state. We will present in Section 3 the MDP-based formulation of the admission control carried out by the SLA Manager component. Once a user requesting a SLA has been admitted by the SLA Manager, it starts generating requests to the composite service managed by MOSES until its contract ends. Each user request involves the invocations of the tasks according to the logic specified by the composite service workflow. For each task invocation, MOSES binds dynamically the task of the abstract composition to an actual implementation (i.e., concrete service), selecting it from the pool of network accessible service providers that offer it. We model this selection by associating with each task S_i a vector $\boldsymbol{x}_i = (\boldsymbol{x}_i^1, ..., \boldsymbol{x}_i^{|K|})$, where $\boldsymbol{x}_i^k = [x_{ij}^k]$ and $i.j \in I_i$. Each entry x_{ij}^k of $\boldsymbol{x}_i^k$ denotes the probability that the class-k request will be bound to concrete service $i.j$. MOSES determines the service selection strategy $\boldsymbol{x}$ by solving the

following revenue maximization problem **MAXRW:**

$$\textbf{max } C(\Lambda, \boldsymbol{x}) = \sum_{k \in K} \Lambda^k \Big[C^k - \Big(C^k(\Lambda, \boldsymbol{x}) + P_\tau^k \tau^k + P_\rho^k \rho^k \Big) \Big]$$

$$\textbf{subject to: } T^k(\Lambda, \boldsymbol{x}) \le T_{\max}^k + \tau^k, \quad k \in K \tag{1}$$

$$R^k(\Lambda, \boldsymbol{x}) \ge R_{\min}^k - \rho^k, \quad k \in K \tag{2}$$

$$C^k(\Lambda, \boldsymbol{x}) \le C^k, \quad k \in K \tag{3}$$

$$l_{ij}(\Lambda, \boldsymbol{x}) \le l_{ij}, \quad j \in I_i, \, i \in \mathcal{F} \tag{4}$$

$$x_{ij}^k \ge 0, j \in I_i, \sum_{j \in I_i} x_{ij}^k = 1, \quad i \in \mathcal{F}, k \in K \tag{5}$$

$$\tau^k \ge 0, \rho^k \ge 0, \quad k \in K \tag{6}$$

where: $\Lambda = (\Lambda^k)_{k \in K}$ and $\Lambda^k = \sum_u L_u^k$ is the aggregate class-k users service request rate; $T^k(\Lambda, \boldsymbol{x})$, $R^k(\Lambda, \boldsymbol{x})$, and $C^k(\Lambda, \boldsymbol{x})$ the class-k response time, reliability and implementation cost, respectively, under the service selection strategy $\boldsymbol{x}$. The objective function $C(\Lambda, \boldsymbol{x})$ is the broker per unit of time reward. It is the sum over all service classes of the service class-k invocation rate Λ^k times the per invocation reward, that is C^k minus the cost $C^k(\Lambda, \boldsymbol{x})$ (which is increased by the penalty $P_\tau^k \tau^k + P_\rho^k \rho^k$ for service violation). For space limitations we omit the details on inequalities (1)-(6), which can be found in [7].

Since the proposed optimization problem is a Linear Programming problem it can be efficiently solved via standard techniques. We will denote by $\boldsymbol{x}^*(\Lambda)$ the optimal service selection policy.

3 An MDP Formulation for MOSES Admission Control

In this section we formulate the MOSES admission control problem as a Continuous-time Markov Decision Process (CTMDP). We first present our broker model and define the user state space model. Then, we define the broker actions/decisions and present the state transition dynamics. Finally, we present our performance criterion and how to compute the optimal policy.

3.1 Model

We consider a broker that has a fixed set of candidate concrete services (and associated SLAs) with which offers the composite service to prospective users. Prospective users contact the broker to establish a SLA for a given class of service k and for a given period of length. We model the arrival process for service class k and contract duration of expected length $1/\mu_d$ as a Poisson process with rate λ_d^k. We assume that the contract durations are exponentially distributed with finite mean $1/\mu_d > 0$, $d \in D = \{1, \ldots, d_{\max}\}$ (which we assume for the sake of simplicity to not depend on the service class k). Upon a user arrival, the broker has to decide whether to admit a user or not. If a user is admitted, the user will

generate a flow of requests at rate L^k for the duration of the contract. When a user contract expires, the user simply leaves the system. The broker set of actions is then just the pair $\mathcal{A} = \{a_a, a_r\}$, with a_a denoting the accepting decision and a_r the refusal decision.

We model the state of our system as in [15]. The state s consists of the following two components:

- the broker users matrix $n = (n_d^k)_{k \in K, d \in D}$, where n_d^k denotes the number of users for each service class k and expected contract duration $1/\mu_d$ before the last random event occurred;
- the last random event ω.

n takes values in the set $\mathcal{N}$ of all possible broker user matrices for which the optimization problem **MAXRW** introduced in Section 2 has a feasible solution. ω represents the last random event, *i.e.*, a user arrival or departure, occurred in the system. We will denote it by a matrix $\omega = (\omega_d^k)_{k \in K, d \in D}$, where $\omega_d^k = 1$ if a new user makes an admission request for service class k and for a contract duration with mean $1/\mu_d$, $\omega_d^k = -1$ if an existing user of class k and contract duration of mean $1/\mu_d$ terminates his contract, and $\omega_d^k = 0$ otherwise. We will denote by Ω the set of all possible events.

The state space $\mathcal{S}$ consists of all possible user configuration-next event combinations, *i.e.*, $\mathcal{S} = \{s = (n, \omega) | n \in \mathcal{N}, \omega \in \Omega, \omega_d^k \geq 0 \text{ if } n_d^k = 0\}$. It is important to observe that, following [15], there is a subtle relationship between a state $s = (n, \omega)$ value and the associated user configuration n. Indeed, if the current state is $s = (n, \omega)$ it means that the user configuration *was* n before the last occurred event ω. The actual current user configuration is instead n', which depends on both the event ω and the decision a taken by the broker as discussed below.

For each state $s = (n, \omega)$, the set of available broker actions/decisions $A(s)$ depends on the event ω. If ω denotes an arrival, the broker has to determine whether to accept it or not; thus $A(s) = \{a_a, a_r\}$. If, instead, ω denotes a contract termination, there is no decision to take and $A(s) = \emptyset$.

System transitions are caused by users arrivals or departures. Given the current state $s = (n, \omega)$, the new state $s' = (n', \omega')$ is determined as follows:

- ω' is the event occurred;
- n' is the user configuration *after* the event ω (the previous event) and the decision $a \in A(s)$ taken by the broker upon ω. n' differs from n upon a user departure or a user arrival provided it is accepted. In compact form we can write $n' = n + \omega \mathbf{1}_{\{a \neq a_r\}}$, where $\mathbf{1}_{\{.\}}$ is the indicator function.

Observe that while the system is in state s the actual user configuration is n', which will characterize the next state s'. Table 1 summarizes all the possible transitions. The associated transition rates are then readily obtained:

$$q_{ss'} = \begin{cases} \lambda_d^k & \omega_d'^k = 1 \\ \mu_d n_d'^k & \omega_d'^k = -1 \end{cases} \tag{7}$$

Table 1. System transitions.

Event ω	Decision	Next state $s' = (n', \omega')$
arrival	admitted ($a = a_a$)	$(n + \omega, \omega')$
	refused ($a = a_r$)	(n, ω')
departure	-	$(n + \omega, \omega')$

An admission control policy π for the service broker is a function $\pi : \mathcal{S} \to \mathcal{A}$ which defines for each state $s \in \mathcal{S}$ whether the broker should admit or refuse a new user. We are interested in determining the admission control policy which maximizes the broker discounted expected reward/profit with discounting rate $\alpha > 0$. For a given policy π let $v_\alpha^\pi(s)$ be the expected infinite-horizon discounted reward given s as initial state, defined as $v_\alpha^\pi(s) = E_s^\pi \left\{ \sum_{i=1}^\infty \int_{\sigma_n}^{\sigma_{n+1}} e^{-\alpha u} c(s_i, a_i) du \right\}$ where $\sigma_1, \sigma_2, \dots$ represents the time of the successive system decision epochs which, in our model, coincide with user arrivals and departures. $c(s_i, a_i)$ is the broker reward between decision epochs i and $i + 1$, that is MOSES reward under the optimal service selection strategy x^* between the two decision epochs. To compute its value, let us denote by $\Lambda^k(s, a)$ the aggregate class-k users service request rate when the state is s and the broker action was a and let $\Lambda(s, a) = (\Lambda^k(s, a))_{k \in K}$. Then, $\Lambda^k(s, a) = n'^k L^k$ where $n' = n + \omega \mathbf{1}_{\{a \neq a_r\}}$ is the next state configuration, given the actual state is $s = (n, \omega)$ and decision a was taken, and $n'^k = \sum_{d \in D} n_d'^k$ is the number of users in service class k. We thus have

$$c(s, a) = C(\Lambda(s, a), x^*(\Lambda(s, a))) \tag{8}$$

The optimal policy π^* satisfies the optimality equation (see 11.5.4 in [13]):

$$v_\alpha^{\pi^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s, a)}{\alpha + \beta(s, a)} + \sum_{s' \in \mathcal{S}} \frac{q_{ss'}}{\alpha + \beta(s, a)} v_\alpha^{\pi^*}(s') \right\}, \forall s \in \mathcal{S} \tag{9}$$

where $\beta(s, a)$ is the rate out of state s if action a is chosen, *i.e.*, $\beta(s, a) = \sum_{k \in K} \sum_{d \in D} (\lambda_d^k + n_d'^k \mu_d)$. In (9), the first term $\frac{c(s,a)}{\alpha + \beta(s,a)}$ represents the expected total discounted reward between the first two decision epochs given the system initially occupied state s and taken decision a. The second term represents the expected discounted reward after the second decision epoch under the optimal policy. The optimal policy π^* can be obtained by solving the optimality equation (9) via standard techniques, *e.g.*, value iteration, LP formulation [13].

A potential limitation of the infinite-horizon approach we presented above arises from the curse of dimensionality which gives rise to state explosion. As shown in the next section, in our setting, even for small problem instances, we incurred high computational costs because of the large state space. As a consequence, this approach might not be feasible for online operation where a new policy must be recomputed as user statistics or the set of concrete services vary over time unless we resort to heuristics. In alternative, we also consider finite horizon policies which not only are amenable to efficient implementations, and allow to trade-off complexity vs horizon length, but also take into account the

fact that in a time varying system it might not be appropriate to consider a stationary, infinite horizon policy. In a finite-horizon setting, our aim is to optimize the expected N step finite-horizon discounted reward given s as initial state, $v_\alpha^{\pi_N}(s)$, which is defined as (3.1) with ∞ replaced by N in the summation, where N defines the number of decision epochs over which the reward is computed.

For finite horizon problem, the optimal policy π_N^* satisfies the following optimality equation:

$$v_{i,\alpha}^{\pi_N^*}(s) = \sup_{a \in A(s)} \left\{ \frac{c(s,a)}{\alpha+\beta(s,a)} + \sum_{s' \in \mathcal{S}} \frac{q_{ss'}}{\alpha+\beta(s,a)} v_{i+1,\alpha}^{\pi_N^*}(s') \right\}, \forall s \in \mathcal{S} \quad (10)$$

where $v_{i,\alpha}^{\pi_N}(s)$ is the expected discounted reward under policy π from decision epoch i up to N and $v_\alpha^{\pi_N}(s) = v_{1,\alpha}^{\pi_N}(s)$. The optimal policy π_N^* can be computed directly from (10) via backward induction by exploiting the recursive nature of the optimality equation [13].

4 Experimental Analysis

In this section, we present the experimental analysis we have conducted through simulation to assess the effectiveness of the MDP-based admission control. We first describe the simulation model and then present the simulation results.

4.1 Simulation Model

Following the broker model in Section 3, we consider an open system model, where new users belonging to a given service class $k \in K$ and expected contract duration $1/\mu_d$ arrive according to a Poisson process of rate λ_d^k. We also assume exponential distributed contract duration. Once a user is admitted, it starts generating requests to the composite service according to an exponential inter-arrival time with rate L_u^k until its contract expires.

The discrete-event simulator has been implemented in C language using the CSIM 20 tool. Multiple independent random number streams have been used for each stochastic model component. The experiments involved a minimum of 10,000 completed requests to the composite service; for each measured mean value the 95% confidence interval has been obtained using the run length control of CSIM. The admission control policies have been implemented in MATLAB.

4.2 Experimental Results

We illustrate the dynamic behavior of our admission control policies assuming that MOSES provides the composite service whose workflow is shown in Figure 1 (right). For the sake of simplicity, we assume that two candidate concrete services (with their respective SLAs) have been identified for each task, except for S_2 for which four concrete services have been identified. The respective SLAs differ in

terms of cost c, reliability r, and response time t (being the latter measured in sec.); the corresponding values are reported in Table 2 (left) (where $i.j$ denotes the concrete service). For all concrete services, $l_{ij} = 10$ invocations per second.

On the user side, we assume a scenario with four classes (*i.e.*, $1 \leq k \leq 4$) of the composite service managed by MOSES. The SLAs negotiated by the users are characterized by a wide range of QoS requirements as listed in Table 2 (right), with users in class 1 having the most stringent performance requirements and highest cost paid to the broker, and users in class 4 the least stringent performance requirements and lowest cost. The penalty rates P_τ^k and P_ρ^k are equal to the reciprocal of the corresponding SLA parameter. Furthermore, for

Table 2. Concrete service (left) and service class (right) SLA parameters.

$i.j$	c_{ij}	r_{ij}	t_{ij}
1.1	6	0.995	2
1.2	3	0.99	4
2.1	4.5	0.99	1
2.2	4	0.99	2
2.3	2	0.95	4

$i.j$	c_{ij}	r_{ij}	t_{ij}
2.4	1	0.95	5
3.1	2	0.995	1
3.2	1.8	0.995	2
4.1	1	0.995	0.5
4.2	0.8	0.99	1

$i.j$	c_{ij}	r_{ij}	t_{ij}
5.1	2	0.99	2
5.2	1.4	0.95	4
6.1	0.5	0.99	1.8
6.2	0.4	0.95	4

Class k	C^k	$R^k_{\min}$	$T^k_{\max}$
1	25	0.95	7
2	18	0.9	11
3	15	0.9	15
4	12	0.85	18

each service class we consider two possible contract durations (*i.e.*, $d_{\max} = 2$), which can be either *short* or *long*. Therefore, the system state $s = (n, \omega)$ is characterized by a 4×2 broker users matrix n, as defined in Section 3.1.

We compare the results of the following admission control policies for MOSES. Under the **infinite horizon** policy, the admission control decisions are based on the optimal policy π^*, which is obtained by solving the optimality equation (9) via the value iteration method setting the discount rate $\alpha = -\ln(0.9) = 0.1054$. With the **1-step horizon** policy, the admission control decisions are based on the optimal policy π_N^* with a local 1-step reasoning, *i.e.*, $N = 1$. In this case, as explained in Section 3, we obtain π_N^* by solving (10). Finally, with the **blind** policy, no reasoning about future rewards is considered, because MOSES accepts a new contract request if the service selection optimization problem **MAXRW** described in Section 2 can be solved given the SLA requested by the new user and the SLAs agreed by MOSES with its currently admitted users.

We consider three different scenarios, where we vary the arrival rate of the contract requests. In all the scenarios the amount of request generated by an admitted user is $L_u^k = 1$ req/sec and the contract duration is fixed to $(1/\mu_d)_{d \in D} = (50, 200)$, where the first component corresponds to short contracts and the latter to longer contracts. In the following, we will denote short and long contracts with s and l, respectively.

In the *first scenario*, we set the matrix $(\lambda_d^k)_{k \in K, d \in D} = \begin{pmatrix} 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \end{pmatrix}$, that is all the contract requests arrive at the same rate, irrespectively of the service class. In the *second scenario*, $(\lambda_d^k)_{k \in K, d \in D} = \begin{pmatrix} 0.02 & 0.02 \\ 0.02 & 0.02 \\ 0.04 & 0.04 \\ 0.08 & 0.08 \end{pmatrix}$, that is contract requests

for service classes 3 and 4 arrive at a double (class 3) or quadruple (class 4) rate with respect to requests for service classes 1 and 2. In the *third scenario*, $(\lambda_d^k)_{k \in K, d \in D} = \begin{pmatrix} 0.08 & 0.08 \\ 0.04 & 0.04 \\ 0.02 & 0.02 \\ 0.02 & 0.02 \end{pmatrix}$, that is contract requests for service classes 1 and 2 arrive at a quadruple (class 1) or double (class 2) rate with respect to requests for service classes 3 and 4. To compare the performance of the different admission

Table 3. Average reward per second.

Admission policy	Scenario 1	Scenario 2	Scenario 3
Blind	40.536	25.012	58.801
1-step horizon	59.607	63.865	75.751
Infinite horizon	66.737	65.553	76.116

control policies, we consider as main metrics the average reward per second of the service broker over the simulation period and the percentage of rejected contract requests. Furthermore, for the MDP-based admission control policies we analyze also the mean execution time. For space reason, we do not show the QoS satisfaction levels achieved by the users for the response time, reliability, and cost SLA parameters. Anyway, we found that once a contract request has been accepted, the QoS levels specified in the SLAs are quite largely met by MOSES for each flow of service class, independently on the admission policy.

Table 3 shows the average reward per second earned by the service broker for the various admission control policies and under the different considered scenarios. As expected, the infinite horizon policy maximizes the broker reward, achieving significant improvements over the blind policy under all scenarios. In these scenarios, the 1-step horizon policy yields results close to the optimum.

We now separately analyze the performance metrics for each scenario. From Table 3 we can see that in the first scenario the 1-step horizon policy let the broker earn 47% more than the blind policy, while the improvement achieved by infinite horizon policy over blind is even higher, being equal to almost 65%.

Figure 4.2 (left) shows the percentage of rejected SLA contracts for all the service classes, distinguishing further between short and long contract durations, achieved by the different admission control policies (for each policy, the first four bars regard the short-term contracts for the various service classes, while the latter four the long-term ones). While the blind policy is not able to differentiate among the service classes from the admission control point of view, the MDP-based policies tend to accept the more profitable classes 1 and 2, which pay more for the composite service, and to reject the less profitable ones.

For the second scenario, which is characterized by a higher contract request arrival rate for classes 3 and 4, Figure 4.2 (right) shows that, as expected, all the admission control policies reject a higher percentage of contract requests for these classes with respect to the first scenario. However, MDP-based admission control, independently on the horizon width, prefers clearly service classes 1 and
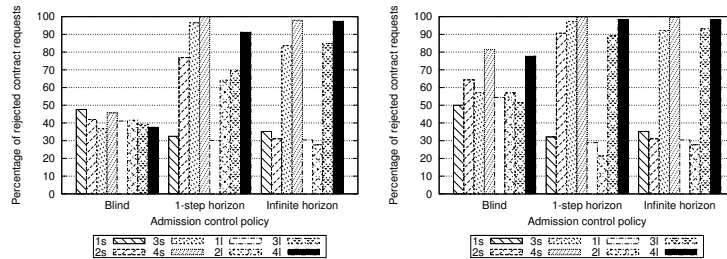
Fig. 2. Rejected contract requests under scenario 1 (left) and 2 (right).

2 with respect to 3 and 4, since the former ones yield higher rewards than the latter which instead experienced high refusal percentages. We also observe that the infinite horizon policy slightly differentiates within classes 1 and 2 according to the contract duration: long-term contracts are preferred to short-term ones (a reduction in the rejection decisions equal to 16% and 11% for long-term classes 1 and 2, respectively). Analyzing the average reward reported in Table 3, we can see that under the second scenario the MDP-based policies allow the broker to more than double its revenue: the 1-step horizon and infinite horizon policies let the broker earn 155% and 162% respectively more than the blind policy. In
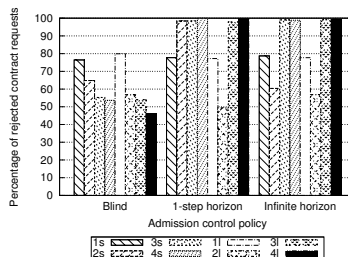

Fig. 3. Rejected contract requests under scenario 3.

the third scenario, where classes 1 and 2 experience higher arrivals rates the MDP-based admission control policies still allow to achieve a good improvement in the reward gained by the broker, as reported in Table 3 (29% and 29.5% for 1-step horizon and infinite horizon policies, respectively, when compared to the blind one). Figure 3 shows the corresponding rejection percentage.

Under all the considered scenarios, the 1-step horizon policy allows the service broker to make a profit comparable, although slightly reduced, to the infinite horizon policy. However, a strong argument in favor of the 1-step horizon policy regards the execution time needed to achieve the optimal decision. We have measured the mean execution time on a machine with Intel Core 2 Duo T7250 2

GHz and 2 GB RAM. The 1-step horizon policy requires only 0.0021 sec, while the infinite horizon one requires 233 sec. for the state space generation, 5502 sec. for the matrix generation, and 800 sec. for the value iteration method. This long execution time is also due to the computation of $c(s, a)$, which requires to solve the service selection optimization problem (see (8)). Therefore, the reduced computational cost of the 1-step horizon policy makes it amenable to take online admission control decisions.

5   Conclusions

In this paper, we have studied the admission control problem for a service broker, MOSES, which offers to prospective users a composite service with different QoS levels. We have formulated the admission control problem as a Markov Decision Process with the goal to maximize the broker discounted reward, while guaranteeing non-functional QoS requirements to its users. We have considered both infinite-horizon and the less computational demanding finite-horizon cost functions. We have compared the different solutions through simulation experiments. Our results show that the MDP-based policies guarantee much higher profit to the broker while guaranteeing the users QoS levels with respect to a simple myopic policy which accepts users as long as the broker has sufficient resources to serve them. In particular, the simple 1-step horizon policy achieves near to optimal performance at a fraction of the computational cost which makes it amenable to online implementation.

In our future work we plan to implement the MDP-based admission control in the existing MOSES prototype and run experiments in realistic scenarios.

References

1. Ardagna, D., Pernici, B.: Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. 33(6), 369–384 (2007)
2. Bannazadeh, H., Leon-Garcia, A.: Online optimization in application admission control for service oriented systems. In: Proc. IEEE APSCC '08 (2008)
3. Beckmann, M., Subramanian, R.: Optimal replacement policy for a redundant system. OR Spectrum 6(1), 47–51 (1984)
4. Bellucci, A., Cardellini, V., Di Valerio, V., Iannucci, S.: A scalable and highly available brokering service for SLA-based composite services. In: Proc. ICSOC '10. LNCS, vol. 6470. Springer (Dec 2010)
5. Bichler, M., Setzer, T.: Admission control for media on demand services. Service Oriented Computing and Applications 1(1), 65–73 (2007)
6. Canfora, G., Di Penta, M., Esposito, R., Villani, M.: A framework for QoS-aware binding and re-binding of composite web services. J. Syst. Softw. 81 (2008)
7. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F.: Flow-based service selection for web service composition supporting multiple QoS classes. In: Proc. IEEE ICWS '07. pp. 743–750 (2007)
8. Chen, M., Feldman, R.: Optimal replacement policies with minimal repair and age-dependent costs. Eur. J. Oper. Res. 98(1), 75–84 (1997)

9. Doshi, P., Goodwin, R., Akkiraju, R., Verma, K.: Dynamic workflow composition: using Markov decision processes. Int'l J. Web Service Res. 2(1) (2005)
10. Gao, A., Yang, D., Tang, S., Zhang, M.: Web service composition using Markov decision processes. In: Proc. WAIM '05. LNCS, vol. 3739. Springer (2005)
11. OASIS: Web Services Business Process Execution Language Version 2.0 (Jan 2007)
12. Pillai, S., Narendra, N.: Optimal replacement policy of services based on Markov decision process. In: Proc. IEEE SCC '09. pp. 176–183 (2009)
13. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic, Dynamic Programming. Wiley (1994)
14. Wang, H., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Proc. ICSOC '10. LNCS, vol. 6470, pp. 92–107. Springer (Dec 2010)
15. Wu, C., Bertsekas, D.: Admission control for wireless networks. Tech. Rep. LIDS-P-2466, Lab. for Information and Decision Systems, MIT (1999)