On Choosing An Efficient Service Selection Mechanism In Dynamic Environments *

Murat Şensoy and Pınar Yolum

Department of Computer Engineering, Boğaziçi University, Bebek, 34342, Istanbul, Turkey,
{murat.sensoy,pinar.yolum}@boun.edu.tr

Abstract. Consumers use service selection mechanisms to decide on a service provider to interact with. Although there are various service selection mechanisms, each mechanism has different strengths and weaknesses for different settings. In this paper, we propose a novel approach for consumers to learn how to choose the most useful service selection mechanism among different alternatives in dynamic environments. In this approach, consumers continuously observe outcomes of different service selection mechanisms. Using their observations and a reinforcement learning algorithm, consumers learn to choose the most useful service selection mechanism with respect to their trade-offs. Through the simulations, we show that not only the consumers choose the most useful service selection mechanism using the proposed approach, but also the performance of the proposed approach does not go below the lower-bound defined by the trade-offs of the consumers.

1 Introduction

Fulfilling transactions with the help of agents is becoming an essential part of e-commerce. A crucial role of agents in such transactions is to help consumers identify the parties that they can interact with [1]. For example, a consumer that wants to buy a product needs to decide which service provider to use for her transaction. The consumer's agent can help the consumer find the right service provider by interacting with other consumers and exchanging relevant information such as their ratings [2] or their experiences [3] with different service providers. The consumer's agent can then use the information it collects to reason and select the appropriate service provider for its needs.

In a real-life implementation of consumer agents, there should be at least one service selection mechanism embedded in the agent. This service selection mechanism is used by the agent to decide on a service provider to interact with. However, which service selection mechanism to implement in an agent is a difficult task for an agent developer for at least two reasons.

– **Trade-offs among different mechanisms.** Each mechanism has its own advantages and disadvantages over the others under different settings. In our previous research, we have observed that if the consumers in the environment have almost

* This research has been partially supported by Boğaziçi University Research Fund under grant BAP06A103 and The Scientific and Technological Research Council of Turkey by a CAREER Award under grant 105E073.

the same taste, providers satisfying one consumer tend to satisfy another consumer, too. Further, rating-based mechanisms are simple and fast as well as effective in terms of achieved satisfaction. On the other hand, if the tastes of consumers are highly different, rating-based mechanisms may have a bad performance in terms of achieved satisfaction [3, 4]. This means that when the environment is static and consumers have similar preferences, rating-based mechanisms may give perfect results in a fast way in such an environment. On the other hand, an experience-based mechanism may be preferable if the consumers' preferences and tastes vary. Hence, under different situations, the same service selection mechanism has highly different utilizations for the same consumers.

– **Trade-offs among different consumer agents.** The expectations of consumers from a service selection mechanism may vary. For example, one consumer may trade off time for performance. That is, the consumer may prefer to find service providers faster even when that means the best service provider will not be found. On the other hand, another consumer may prefer the slower mechanism because of the fact that its achieved satisfaction is higher.

Ideally, the consumer agent should be able to choose the service selection mechanism to use based on its current trade-offs as well as its observation of the environment. To accomplish this, we propose agents to learn the pros and cons of service selection mechanisms to use in dynamic environments and to decide on the mechanism at run time. As a learning technique, we employ reinforcement learning (RL) since it allows agents to learn from their actions in the environment [5]. Through simulations, we show that using our proposed approach, consumers can successfully learn to select the most useful mechanisms for service selection under different configurations of the environment.

The rest of this paper is organized as follows: Section 2 explains the proposed approach, with necessary background knowledge. Section 3 gives a brief overview of service selection mechanisms that are used in this work and provides their performance evaluation. Section 4 provides our experimental results. Finally, Section 5 summarizes our contribution and compares it to relevant literature.

2 Learning To Choose Among Service Selection Mechanisms

In real-life, given a number of service selection mechanisms, we expect service consumers to pick a mechanism that is most suitable for their current situation. For example, if the consumer is in a hurry, it might prefer a service selection mechanism that will find it a service provider quickly; whereas at other times, the quality of the found service provider may be more important. Making choices between service selection mechanisms would be trivial if agents could observe and characterize their environment perfectly. However, this is rarely the case. Most often, agents are not aware of the service demands of other agents, those agents' past experiences, their expectations and so on. Interestingly, the performance of most service selection mechanisms (time and quality-wise) are dependent of these characteristics of the environment. Therefore, we need to provide consumer agents with means to detect which mechanism to use in their environment to satisfy QoS constraints such as the time required for service selection or desired ratio of satisfaction.

In different configurations of the environment, a service selection mechanism may have different utilizations for a consumer. Therefore, if the agent finds out that its environment is changing, then the consumer should switch to another service selection mechanism that has better utilization for that specific setting. For this purpose, the consumer should continuously observe the environment and should learn the best service selection mechanism for itself under different configurations of the environment. This intuition becomes more concrete when we analyze the different parameters of an environment.

2.1 Environment Characteristics

There are three important parameters related to service selection [3, 4, 6].

Variations in service demand: Each service consumer changes its demand characteristics after receiving a service with a probability of P_{CD}.

Variations in service satisfaction: Satisfaction criteria of different consumers may vary significantly. Even though two consumers have similar or the same service demands, their degree of satisfaction for the same supplied service may be highly different. To formulate this, we use the parameter β. This parameter represents the ratio of the consumers having similar demands but conflicting satisfaction criteria. For example, a consumer C has a service demand and there are 40 consumers having a demand similar to that of the consumer C. In the case of $\beta = 0.1$, we expect that 10% of the 40 consumers get dissatisfied with the supplied services that can satisfy the consumer C.

Variations in service quality: Some providers may supply marginally different services at different points of time for the same service demand and conditions. To simulate this, with a very small probability, providers deviate from their expected behavior in the simulations [3, 4]. This probability is called probability of indeterminism (PI). Think of a provider who usually produces unsatisfactory services for a specific service demand. If this provider produces a perfect service for this service demand in a transaction with a consumer, it may mislead the consumers in their future decisions.

The variations in the service demand, service satisfaction, and service quality create significant trade-offs between service selection mechanisms.

2.2 Reinforcement Learning

Reinforcement learning (RL) [5] is an ideal learning technique to enable agents to learn the environment and thus decide on which strategy to use in that particular situation. That is why we propose to use RL for choosing a service selection mechanism in dynamic environments.

Let us review the basic structure of an RL algorithm. In RL, there is a learning agent and an environment that is observed by this agent. The environment has a set of discrete states. At each state, there exists a set of available actions for the agent. When the agent takes an action in a state, it receives a reward in turn and also the observed state of the environment may change as a result of this action. The action to be taken in each state is determined by the policy adopted by the agent. After a set of trial-and-error interactions with the environment, the agent learns an optimal policy for choosing the best action in a given state of the environment so that the total reward is maximized [5].

In RL, the environment is defined by a finite set of states $S = \{s_1, s_2, \ldots, s_N\}$, and the agent has a finite set of actions $A = \{a_1, a_2, \ldots, a_M\}$. In this setting, the agent observes the state of the environment $s_t \in S$ in time t, and chooses an action $a_t \in A$, and receives a scalar reward r_{t+1} after executing a_t. Each state has a value in terms of the maximum discounted reward expected in that state. Value of a state is formulated in Equation 1. Thus, the purpose of RL is to construct an optimal action policy that maximizes the expected discounted reward in each state. In the equation, γ, $(0 < \gamma < 1)$, is a discount factor so that distant rewards are less important. The equation expresses that expected value of a state, s_t, is the weighted sum of the rewards received when starting in the state s_t and following the current policy.

$$V(s_t) = E\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+1}\right] \tag{1}$$

The action selection at each step is based on Q-values, which are related to the goodness of the actions. The Q-value, $Q(s, a)$, is the total discounted reward that the agent would receive when it starts at a state s, performs an action a, and behaves optimally thereafter. The Q-values can be estimated by Temporal Difference Learning (TD) [7] methods such as Q-Learning [8] and SARSA [5]. In this work, we choose to use SARSA reinforcement learning algorithm [5](shown in Algorithm 1). SARSA has two important advantages compared to other approaches [9]. First, it has better convergence guarantees compared to other RL approaches such as Q-learning. Secondly, it learns much more rapidly and in the early part of learning, its average policy is better than that of the other RL approaches.

SARSA Algorithm In Algorithm 1, initial Q-values are set to zero and the current state s is determined (Line 1). While the current state is not the terminal state of the agent (Line 2-9), the agent selects an action a using a policy (π) derived from the Q-values (Line 3). There may be different ways of deriving a policy from Q-values. One of the most popular approaches is $\epsilon - greedy$. In this approach, with probability ϵ, we choose an action uniformly randomly among all possible actions. This is called exploration. As well as, with probability $1 - \epsilon$, we choose the action having the maximum Q-value ($\pi(s) = max_{a_i} Q(s, a_i)$) and this is called exploitation. We do not want to explore indefinitely, so we start with a high ϵ value and continuously decrease it at each time step (e.g., $\epsilon = 1/t$). After determining a using π, a is executed (Line 3). As a result an immediate reward is taken and a new state is observed s' (Line 4). SARSA algorithm is different from other TD approaches such as Q-Learning in a way that it uses the current policy to update Q-values. Therefore, in the algorithm, the action that should be taken in state s' is also determined using the policy π (Line 5). Then, s, a, r, s', and a' are used to update $Q(s, a)$ (Line 7). In the update formula of Q-values, γ is the discount factor and η is the learning rate ($0 < \eta < 1$). Discount factor is used to weight immediate rewards more heavily than future rewards. The closer γ is to 1 the greater the weight of future rewards. The learning rate controls how much weight we give to the recent reward, as opposed to the old Q-value estimate. We typically start with a high learning rate and lower the learning rate as time progresses. After updating Q-value, s and a are updated properly for the next time step (Line 8).

Algorithm 1 [5]

1: Initialization: initialize all Q(s,a) value to zero and observe the current state s
2: **while** (s is not terminal state) **do**
3: Select an action a using policy derived from Q (e.g., using $\epsilon - greedy$) and execute it.
4: Observe immediate reward r and the new state s'
5: Choose the action a' in state s' using policy derived from Q (e.g., using $\epsilon - greedy$)
6: Update $Q(s, a)$:
7: $Q(s, a) \leftarrow Q(s, a) + \eta \times (r + \gamma \times Q(s', a') - Q(s, a))$
8: $s \leftarrow s'$, $a \leftarrow a'$
9: **end while**

Reward Function The reward function (r) that we use with SARSA algorithm is given in Equation 2. In the equation, $E(R_{action})$ and $E(T_{action})$ denote the expected ratio of satisfaction and the expected time required for the chosen action, respectively. This reward function reflects the trade-offs of the service consumers. According to the reward function, a consumer is given a negative reward after choosing an action if there is another action with an expected ratio of satisfaction that is at least 10% better than that of the chosen action. Even though there is no such action, the consumer gets a negative reward again if the chosen action is at least 10% slower than another action whose ratio of satisfaction is at most 1% worse than that of the chosen action. If none of these cases occurs, the agent gets a positive reward. Using this reward function, consumers learn to choose fast service selection strategies without trading off the ratio of satisfaction more than 10%. During its life-time, the consumer continuously computes and revises $E(R_{action})$ and $E(T_{action})$. For each action, the consumer records the time required for the action and the result of the action in terms of the degree of satisfaction. By simply averaging these values over time, the consumer computes $E(T_{action})$ and $E(R_{action})$, respectively. In order to handle highly dynamic environments, the consumer uses only the recent information in its computations (i.e., information from the last 20 time steps).

$$r(act) = \begin{cases} -1 \text{ if } \exists X | \frac{(E(R_X) - E(R_{act}))}{E(R_{act})} > .1 \\ -1 \text{ if } \exists X | abs(\frac{(E(R_X) - E(R_{act}))}{E(R_{act})}) < .01 \wedge \frac{(E(T_{act}) - E(T_X)))}{E(T_X)} > .1 \\ +1 \text{ otherwise} \end{cases} \quad (2)$$

2.3 Discretization of Continuous State Space

In our setting of RL, states of the environment are different configurations of the environment and the actions are different service selection strategies. We can model the states of the environment using the P_{CD}, β and PI parameters. However, agents cannot observe these parameters directly, but they can observe the result of their actions. Intuitively, there is a correlation between the performance of the service selection strategies and these parameters. This intuition enables us to describe states of the environment using the expected ratio of satisfaction of the different service selection strategies, instead of the parameters like P_{CD}, β, or PI. For example, if we consider two service selection

mechanisms A and B, a state can be characterized by a consumer using the expected success of A and B. Therefore, the consumer observes the states of the environment as the pairs of different $E(R_A)$ and $E(R_B)$ values. However, this state space is continuous and needs to be converted into a discrete state space for RL.

By dividing continuous state space to discrete states, we are trying to map different configurations of the environment to a number of discrete states. If the number of discrete states is too small, then highly different configurations of the environment will be mapped to the same discrete state. This will decrease the accuracy of the RL algorithm considerably. If the number of states is set to a number that is much bigger than the optimal number of states, then it will take too much time to train the RL algorithm and at the end, some states will have almost identical Q-values, because of the redundancy of the states. To tackle this problem, we propose to decide on the number of discrete states dynamically. Initially, we start with only one discrete state. We continuously observe the environment and try to estimate the possible states that the environment may be in. Then, we group these possible states into clusters by using *k-means* clustering algorithm. Each cluster is represented by one discrete state. If any cluster of the possible states is too diverse to be represented by only one discrete state, then this cluster is divided into two clusters and a new discrete state is created to represent the new cluster of possible states. Q-values of the discrete state representing the divided cluster is used as the initial estimates of the new state's Q-values. This way, we incrementally increase the number of discrete states only if it is required. This approach has several advantages. First, it prohibits the introduction of redundant states. Second, this approach does not require the knowledge of maximum number of states. Third, this approach increases the convergence rate of Q-values by incrementally introducing the new states and by using pre-computed Q-values as the initial estimates of these states' Q-values. Algorithm 2 presents a modified version of SARSA algorithm with the proposed discretization of continuous state space.

$$wcv_i = \frac{\sum_{L \in c_i} \left[\sum_{j=0}^{n-1} |L[j] - center_i[j]| \right]}{|c_i|} \tag{3}$$

In this algorithm, every state has a label. This label is an n-tuple, where n is the number of available service selection mechanisms and each element of the n-tuple is of $(E(R_{X_i})$: the expected ratio of satisfaction of the service selection mechanism X_i. Initially we have only one state and its label is set to $(0, \ldots, 0)$ (Line 1). During the life time of the agent, it continuously computes expected ratio of satisfaction for each action and creates a label $Label_t$ at time t using computed $E(R_{X_i})$ values (Lines 5–6). $Label_t$ is added to *state-space set* (Line 7). This set is initiated as an empty set (Line 1) and populated with $Label_t$ at each time step. Labels of the states and $Label_t$ are used to determine the next state after taking an action. After taking an action, the distance between computed $Label_t$ and the labels of the states are compared. The state having a label closest to $Label_t$ is set as the current state, s' (Line 9). With a probability P_{update}, we update state labels by running k-means algorithm [10] on state-space set. We initiate k-means algorithm with k centers (k is the number of states). These initial centers are the current labels of the states (Line 15). New clusters and cluster centers are defined after running k-means algorithm. Each state s_i has a corresponding cluster

c_i. We compute within-cluster variance, (wcv_i) for each cluster c_i using the center of the cluster and the labels within the cluster (see Equation 3). If, for any cluster c_i, wcv_i exceeds a predefined threshold (Line 16), a new state is created (s_{new}) (Line 17). One of the labels in c_i is set as the label of s_{new} (Line 18) and Q-values of s_i is copied to s_{new}'s Q-values (Line 19). After creation of new states, we run k-means algorithm again (Line 21). This procedure is repeated until wcv for each cluster is below the threshold. Then, we set the center of each cluster as the new label of the corresponding state (Lines 23–25). Using this approach, we increase the number of states only it is required. Furthermore, increasing the number of states does not affect the performance of the RL algorithm much, because we initialize Q-values of new states with the Q-values of the most similar states. As time advances, Q-values of new states are differentiated by SARSA algorithm.

Algorithm 2

1: Initialization: Create s_{init} with Label $\{0, \ldots, 0\}$, initialize all $Q(s_{init}, a)$ values to zero, $States \leftarrow \{s_{init}\}$, $StateSpace \leftarrow \{\}$
2: $s = s_{init}$
3: **while** (alive()) **do**
4: Select an action a in state s using policy derived from Q, and execute it
5: Update $E(R_{X_i})$ for all of n mechanisms using the data in the last N time steps
6: $Label_t \leftarrow \left\{E(R_{X_0}), \ldots, E(R_{X_{(n-1)}})\right\}$
7: add $Label_t$ to $StateSpace$
8: $r \leftarrow getReward()$
9: $s' \leftarrow min_{s_i} \sum_{j=0}^{n-1} |s_i.Label[j] - Label_t[j]|$
10: Choose the action a' in state s' using policy derived from Q
11: Update $Q(s, a)$:
12: $Q(s, a) \leftarrow Q(s, a) + \eta \times (r + \gamma \times Q(s', a') - Q(s, a))$
13: $s \leftarrow s'$, $a \leftarrow a'$
14: **if** ($Rand() < P_{update}$) **then**
15: KMeans(StateSpace,States)
16: **while** ($\exists c_i | wcv_i > Threshold$) **do**
17: Create new state s_{new}
18: select one label from c_i and assign it as the Label of s_{new}
19: $s_{new}.QValues \leftarrow s_i.QValues$
20: add s_{new} to $States$
21: KMeans(StateSpace,States)
22: **end while**
23: **for** each $s_i \in States$ **do**
24: $s_i.Label \leftarrow c_i.center$
25: **end for**
26: **end if**
27: **end while**

3 Service Selection Mechanisms

To evaluate our proposed approach, we perform simulations using four service selection mechanisms that the agents can choose from at run time. The first two mechanisms are rating-based and the second two mechanisms are experience-based.

3.1 Rating-Based Mechanisms

For a new service demand, a service consumer can select a service provider using ratings from other consumers. A rating is a numerical evaluation (usually between 0 and 1) of a service provider. Ratings are easy to exchange. On the other hand, it is important that ratings are evaluated within their scope. Scope of a rating is the context in which the rater has experienced the service. The two rating-based mechanisms below mainly differ in representing context information with ratings.

- **Selective Ratings (SPS_{SR}):** In this mechanism, a consumer requests ratings only from those consumers who have similar demands with respect to similarity criteria of the consumer. For example, if a consumer wants to buy a bicycle, she asks ratings from other consumers who have bought a bicycle in the past.
- **Context-Aware Ratings (SPS_{CAR}):** A consumer may evaluate the same service provider differently in different contexts. To deal with this, we have previously proposed to enrich the rating information with the context information using an ontology to represent the context [6]. For each service demand in the past, consumers keep a *context-aware rating*. When a service consumer needs a new service, it first defines the context of its service demand using context ontology, and then propagates this definition to its neighborhood. Upon receiving this definition, the neighbors return their context-aware ratings related to a similar context. Afterwards, the consumer aggregates received ratings considering the similarity between their contexts and the context of its service demand.

3.2 Experience-Based Mechanisms

By using context-aware ratings, consumers can explicitly express the scope of ratings. However, ratings still reflect the subjective opinion of the raters. In order to minimize the subjectiveness of rating-based mechanisms, experience-based mechanisms are proposed [3, 4]. In these service selection mechanisms, experience of a consumer with a provider is represented using an experience ontology. This representation contains the requested service and the supplied service in detail. Actually, an experience expresses the story between the consumer and the provider regarding a specific service demand. So, any consumer receiving an experience can evaluate the service provider according to its own criteria using the detailed data in the experience. This approach removes the subjectiveness of the rating-based mechanisms [3].

In experience-based mechanisms, consumers collect experiences instead of ratings. In order to collect experiences, a consumer defines what kind of experiences it is interested in and propagates this definition over its neighborhood. When a neighbor receives this definition, it returns its related experiences to the consumer. The consumer can use

different reasoning techniques to make sense of the information it collects. We analyze two versions: a parametric classification method using Gaussian Model (GM), and a non-parametric method using case-based reasoning (CBR).

- **Using Gaussian Model (SPS_{GM}):** In this approach, a service consumer models each service provider by building a multidimensional Gaussian model using the collected experience data [3]. Demand information in each experience is represented as a vector. Each field in this vector is extracted from the experience ontology. These fields may correspond to property values in experience ontology such as service price. Then, supplied service for this demand is classified as satisfied or dissatisfied with respect to satisfaction criteria of the consumer. The (vector, class) pairs are used as training set for building models of the providers. Then, for each of the models, a discriminant function is defined to compute the probability of satisfaction [10]. The service consumer performs this computation for every service provider and chooses the provider with the highest satisfaction probability.
- **Using Case-Based Reasoning (SPS_{CBR}):** In this approach, a consumer regards each of the collected experiences as a case and gives a score for each experience. Computation of this score depends on several factors: the similarity between the demand within the experience and the current demand of the consumer, desirability of the supplied service within the experience and lastly the age of the experience. After computing a score for each experience, the consumer picks the experience with the highest score and selects the provider supplying the service within this experience. This approach depends on the assumption that if a provider satisfies a service demand which is very similar to or the same as the current demand of a consumer, the provider will probably satisfy the consumer's demand, too.

3.3 Performance Comparisons of Service Selection Mechanisms

We review performances of the aforementioned service selection mechanisms in terms of achieved satisfaction [3, 4, 6] and present their performances in terms of time requirements. For this purpose, we conducted various simulations. The simulation environment is setup with 20 service providers and 400 service consumers. Simulations are run for 100 epochs. In our simulations, there are two metrics to evaluate each strategy. Our first metric is the average ratio of decisions resulted in satisfaction (R), which is called as the *ratio of satisfaction* in short. This metric measures the performance of a service selection mechanism in terms of the achieved satisfaction. For example, $R_{SR} = 0.6$ means that, on the average, only 60% of the service decisions results in satisfaction if SPS_{SR} is used. Our second metric is the average time required for selecting a service provider (T).

For different values of the parameters P_{CD}, β, and PI, Table 1 and Table 2 show the performance of each mechanism in terms of the achived satisfaction and time requirements, respectively. Table 1 reveals that service decisions of a consumer usually results in satisfaction of the consumer if the consumer uses SPS_{GM}. Moreover, the performance of SPS_{GM} in terms of ratio of satisfaction does not change much with different values of P_{CD}, β, and PI. On the other hand, as shown in Table 2, time requirements of SPS_{GM} is much more than that of other mechanisms. Performance of each service

selection mechanism is equally good in terms of the achieved satisfaction when P_{CD}, β, and PI are set to zero. However, increasing P_{CD} results in a serious decrease in the performance of SPS_{SR} in terms of the ratio of satisfaction. Performance of the mechanisms other than SPS_{SR} is not affected by P_{CD} considerably. Performances of SPS_{SR} and SPS_{CAR} in terms of the ratio of satisfaction decrease considerable as β increases. SPS_{SR} is faster than SPS_{CAR} in all cases, because SPS_{CAR} requires reasoning of the context information. Unlike the experience-based mechanisms, the rating-based mechanisms achieve a low ratio of satisfaction as β increases [3], because satisfaction criteria of the consumers get more differentiated as β increases. Table 2 reveals that time requirements of the rating-based mechanisms are much less than that of the experience-based mechanisms, because aggregation of experiences is much more costly than aggregation of ratings in terms of time. Performance of SPS_{CBR} in terms of the ratio of satisfaction is quite well for $PI = 0$ and does not change significantly when P_{CD} or β changes. Furthermore, in terms of time performance, SPS_{CBR} outperforms SPS_{GM}, because the modeling service providers using experiences as in SPS_{GM} requires much more time than examining experiences one by one as in SPS_{CBR}. On the other hand, the performance of SPS_{CBR} in terms of the ratio of satisfaction significantly decreases if indeterminism is observed ($PI > 0$).

In summary, SPS_{GM} is the best service selection mechanism when the ratio of satisfaction is concerned. Unlike the performance of SPS_{GM}, performance of other mechanisms in terms of the ratio of satisfaction is badly affected as P_{CD}, β, or PI increases. P_{CD} affects the performance of SPS_{SR}, β affects the performances of SPS_{SR} and SPS_{CAR}, and lastly PI affects the performance of SPS_{CBR}. Order of the mechanisms from the fastest one to the slowest one is SPS_{SR}, SPS_{CAR}, SPS_{CBR}, and SPS_{GM}. This order does not change as P_{CD}, β, or PI changes.

Table 1. The ratio of service selection decisions resulted in satisfaction for different mechanisms.

P_{CD}	β	PI	R_{SR}	R_{CAR}	R_{GM}	R_{CBR}
0	0	0	0.9682	0.9708	0.9708	0.9708
0.2	0	0	0.6028	0.9730	0.9730	0.9730
0	0.5	0	0.5147	0.7813	0.9666	0.9666
0.2	0.5	0	0.3216	0.4970	0.9710	0.9710
0	0	0.01	0.9734	0.9759	0.9757	0.6635
0.2	0	0.01	0.6172	0.9767	0.9767	0.6046
0	0.5	0.01	0.5196	0.7359	0.9684	0.4768
0.2	0.5	0.01	0.3221	0.5161	0.9719	0.4877

4 Experimental Evaluation

In this section, we evaluate our approach for choosing the most suitable service selection mechanisms in different configurations of the environment using simulations. Simulation settings are explained in the previous section. In the simulations, there are

Table 2. Average time required for service selection mechanisms in milliseconds.

P_{CD}	β	PI	T_{SR}	T_{CAR}	T_{GM}	T_{CBR}
0	0	0	0.89	7.89	1107.74	363.10
0.2	0	0	1.22	13.13	3540.71	588.71
0	0.5	0	0.90	6.01	902.54	247.10
0.2	0.5	0	1.06	12.97	3575.03	581.66
0	0	0.01	0.73	7.30	1108.58	304.64
0.2	0	0.01	0.85	15.90	4566.01	723.28
0	0.5	0.01	0.74	7.81	1269.63	321.99
0.2	0.5	0.01	0.78	14.14	3393.09	850.16

400 consumer agents. Each consumer agent is given four different service selection mechanisms, which are shortly described in Section 3. These consumers use the proposed approach to learn how to choose the most profitable service selection mechanism in their environment. In order to make our evaluations easily understandable, we assume that consumers have the same trade-offs between the time requirements and ratio of satisfaction. These trade-offs are embedded in the reward function that is explained in Section 2.2.

For different configuration of the environment, actions of a consumer are rewarded differently. For example, in case of $\beta = 0$ and $P_{CD} = 0$, a consumer should choose SPS_{SR}. In this configuration of the environment, SPS_{SR} is much faster than other mechanisms, even though performance of other mechanisms in terms of ratio of satisfaction is not significantly better than that of SPS_{SR}. Therefore, the consumer gets a positive reward if it chooses SPS_{SR}; otherwise it gets a negative reward. However, if the value of P_{CD} increases to 0.2, performance of SPS_{SR} in terms of ratio of satisfaction decreases significantly. In this case, SPS_{SR} is still much faster than other mechanisms, but there are other mechanisms that have a ratio of satisfaction at least 10% better than that of SPS_{SR} (see Table 1 and Table 2). Therefore, choosing SPS_{SR} results in a negative reward.

The best of the four service selection mechanisms in terms of the ratio of satisfaction is SPS_{GM}. Hence, we measure the performance of our approach in terms of the performance of SPS_{GM}. When a consumer has a service demand, it uses the proposed approach to chooses a service selection mechanism using the computed policy as explained in Section 2. Then, the consumer selects a service provider using the chosen mechanism. We measure the average ratio of satisfaction (R_{RL}) and the average time required for service selection (T_{RL}) when the proposed approach is used. We also measure the average ratio of satisfaction (R_{GM}) and the average time required for service selection (T_{GM}) if the consumers had used only the SPS_{GM} in their service selections.

Table 3 shows the results of our simulations. The first three columns refer to the parameters that are used to configure the simulation environment. Next two columns are the average ratio of satisfaction for SPS_{GM} and the proposed RL approach, respectively. The last column is the ratio of T_{GM} to T_{RL}. In the first eight rows, there is only one environment configuration during the simulations. On the other hand, in the last

row, environment gradually changes starting from the first configuration to the eighth configuration during the simulation.

Table 3. Performance of Reinforcement Learning in selecting right service selection strategies.

Configuration ID	P_{CD}	β	PI	R_{GM}	R_{RL}	T_{GM}/T_{RL}
1	0	0	0	0.9708	0.9705	114.6
2	0.2	0	0	0.9730	0.9420	46.24
3	0	0.5	0	0.9666	0.9293	14.51
4	0.2	0.5	0	0.9710	0.9185	6.14
5	0	0	0.01	0.9757	0.9706	94.73
6	0.2	0	0.01	0.9767	0.9223	48.75
7	0	0.5	0.01	0.9684	0.9194	2.98
8	0.2	0.5	0.01	0.9719	0.8761	1.12
9	0-0.2	0-0.5	0-0.01	0.9731	0.9213	32.17

In the first configuration of the environment ($P_{CD} = 0$, $\beta = 0$, $PI = 0$), all of the service selection mechanisms have almost the same ratio of satisfaction. Therefore, the proposed approach chooses SPS_{SR} and SPS_{CAR} most of the time, because performance of these two mechanisms are very good in terms of both the achieved satisfaction and the time required for service selection. In this setting, using the proposed approach, the consumers can make as satisfactory service selections as they do using SPS_{GM}. Moreover, using the proposed approach the consumers make service selections 114.6 times faster than they do using SPS_{GM}.

In the second configuration of the environment ($P_{CD} = 0.2$, $\beta = 0$, $PI = 0$), SPS_{SR} is not as good as other strategies any more. Hence, the proposed approach does not prefer SPS_{SR}. Although SPS_{CAR}, SPS_{CBR} and SPS_{GM} have almost the same performance in terms of achieved satisfaction, the proposed approach chooses SPS_{CAR} most of the time. This is due to the fact that SPS_{CAR} is much faster than SPS_{CBR} and SPS_{GM}. As a result, using the proposed approach, consumers select satisfactory services in 94.2% of their service selections. This is achieved in a way that 46.24 times faster than SPS_{GM}.

In the third and fourth configurations, performances of SPS_{SR} and SPS_{CAR} are worse than that of SPS_{CBR} or SPS_{GM} in terms of achieved satisfaction, because both of them are badly influenced as β increases. As a result, the proposed approach chooses either SPS_{CBR} or SPS_{GM} in service selection. The primary choice of our approach is SPS_{CBR}, because it is much faster than SPS_{GM}. As a result, the proposed approach achieves satisfaction in 93% and 92% of the cases in a way 14.52 and 6.14 times faster than SPS_{GM}, respectively for these two configurations.

In the fifth configuration ($P_{CD} = 0$, $\beta = 0$, $PI = 0.01$), SPS_{CBR} has a low ratio of satisfaction, because SPS_{CBR} is badly influenced as PI increases. Therefore, our approach chooses between SPS_{SR}, SPS_{CAR} and SPS_{GM}, which have almost the same ratio of satisfaction in this setting. Most of the time, SPS_{SR} is chosen by our approach, because it is much faster than SPS_{CAR} and SPS_{GM}. As a result, the

proposed approach achieves almost the same ratio of satisfaction as SPS_{GM} does in a way 94.73 times faster than SPS_{GM}.

In the sixth configuration of the environment ($P_{CD} = 0.2$, $\beta = 0$, $PI = 0.01$), only SPS_{CAR} and SPS_{GM} have a desirable performance in terms of the ratio of satisfaction. In this case, the proposed approach chooses the faster mechanism, SPS_{CAR}, rather than SPS_{GM}. Hence, the proposed approach achieves satisfaction in 92% of the cases. In this setting, the proposed approach results in service selections that are 48.75 times faster than that of SPS_{GM}.

In the seventh and eighth configurations, the performance of SPS_{SR}, SPS_{CAR} and SPS_{CBR} are much worse than that of SPS_{GM} in terms of the ratio of satisfaction. As a result, our approach chooses mostly SPS_{GM}. In these settings, the proposed approach achieves satisfaction in 91.9% and 87.6% of the cases, respectively for these two configurations.

We lastly conduct simulations in which configuration of the environment is changed from the first configuration to the eighth configuration. In this challenging case, using the proposed approach, consumers select satisfactory services in 92.1% of their service selections in a way 32.17 times faster than SPS_{GM}.

The worst performance of the proposed approach in terms of achieved satisfaction is in the eighth configuration. In this case, R_{GM} is around 10% better than R_{RL}. In any configuration of the environment, performance of any service selection mechanism in terms of the ratio of satisfaction does not exceed R_{RL} more than 10%. This is the trade-off exactly stated in the reward function. Moreover, we have confirmed this argument by running simulations for different trade-off values.

5 Discussion

Service selection mechanisms vary in their success rate as well as time requirements. Existence of different service selection mechanisms gives rise to a major problem: how to choose the most appropriate service selection mechanism among a range of alternatives. This problem is amplified when the environment is dynamic. Our proposed approach allows agents to learn how to choose the most useful service selection mechanism among different alternatives in dynamic environments. The proposed approach examines the environment and learns to differentiate between different service selection mechanisms with respect to the trade-offs of the consumers. Our experimental results show that consumers choose the most useful service selection mechanism using the proposed approach. The performance of the proposed approach does not go below the lower-bound defined by the trade-offs of the consumers. Even in the case that the environment changes dramatically, the proposed approach exhibits a good performance in terms of both the ratio of satisfaction and the time required for service selection.

To the best of our knowledge there is no approach for choosing one selection mechanism among different alternatives. However, there are various service selection mechanisms proposed in the literature. Yolum and Singh study properties of referral networks for service selection, where referrals are used among the service consumers to locate the service providers [1]. Like the other applications of referral networks, their approach rely on exchanging ratings. Sen and Sajja [11] develop a reputation-based trust model

that is used for selecting processor agents for processor tasks. Each processor agent can vary its performance over time. Agents are looking for processor agents to send their tasks to using only evidence from others. Sen and Sajja propose a probabilistic algorithm to guarantee finding a trustworthy processor. These approaches are stand alone approaches and do not concern choosing a useful service selection mechanism among different alternatives as we do in this work.

Techniques for combining different classifier exist in the literature. Ortega *et. al.* propose an approach that takes a collection of existing classifiers and learning algorithms and creates a combined classifier that takes advantage of all of these classifiers [12]. The basic idea of their approach is that each classifier has a particular subdomain for which it is most reliable. By combining different classifiers, Ortega *et. al.* create a classifier that is reliable for a range of different sub-domains. Although this type of approaches are similar to our approach in a sense that they combine different mechanisms to come up with a better one, they are specifically proposed for the classification problems and they cannot be used directly in the service selection problem that we are addressing in this paper.

As a future work, we plan to enable online addition of new service selection mechanisms to the proposed approach. We also want to add our work a social dimension so that the consumers can recommend each other new service selection mechanisms. Then, the recommended mechanisms can be added to the learning algorithm on the fly.

References

1. Yolum, P., Singh, M.P.: Engineering self-organizing referral networks for trustworthy service selection. IEEE Transactions on Systems, Man, and Cybernetics **A35** (2005) 396–407
2. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS). (2002) 475–482
3. Şensoy, M., Yolum, P.: A context-aware approach for service selection using ontologies. In: Proceedings of Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS). (2006) 931–938
4. Şensoy, M., Yolum, P.: A comparative study of reasoning techniques for service selection. In: Proceedings of the Fifth International Workshop on Agents and Peer-to-Peer Computing (AP2PC). (2006) 91–102
5. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
6. Şensoy, M., Yolum, P.: Ontology-based service representation and selection. Conditionally accepted by IEEE Transactions on Knowledge and Data Engineering (2007)
7. Tesauro, G.: Temporal difference learning and td-gammon. Commun. ACM **38** (1995) 58–68
8. Watkins, C.J.C.H., Dayan, P.: Q-learning. Machine Learning **8** (1992) 279–292
9. Whiteson, S., Taylor, M.E., Stone, P.: Empirical studies in action selection for reinforcement learning. Adaptive Behavior **15** (2007) To appear.
10. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley and Sons (2001)
11. Sen, S., Sajja, N.: Robustness of reputation-based trust: Boolean case. In: Proceedings of the 1st International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS). (2002) 288–293
12. Ortega, J., Koppel, M., Argamon, S.: Arbitrating among competing classifiers using learned referees. Knowledge and Information Systems **3** (2001) 470–490