Multi-Objective Service Composition Using Reinforcement Learning

Ahmed Moustafa and Minjie Zhang

School of Computer Science and Software Engineering
University of Wollongong, Gwnneville, NSW 2500, Australia
{aase995,minjie}@uowmail.edu.au
http://www.uow.edu.au

Abstract. Web services have the potential to offer the enterprises with the ability to compose internal and external business services in order to accomplish complex processes. Service composition then becomes an increasingly challenging issue when complex and critical applications are built upon services with different *QoS* criteria. However, most of the existing *QoS*-aware compositions are simply based on the assumption that multiple criteria, no matter whether these multiple criteria are conflicting or not, can be combined into a single criterion to be optimized, according to some utility functions. In practice, this can be very difficult as utility functions or weights are not well known a priori. In this paper, a novel multi-objective approach is proposed to handle *QoS*-aware Web service composition with conflicting objectives and various restrictions on quality matrices. The proposed approach uses reinforcement learning to deal with the uncertainty characteristic inherent in open and decentralized environments. Experimental results reveal the ability of the proposed approach to find a set of Pareto optimal solutions, which have the equivalent quality to satisfy multiple *QoS*-objectives with different user preferences.

Keywords: Web services, multi-objective optimization, reinforcement learning.

1 Introduction

Web service composition is an important and effective technique that enables individual services to be combined together to generate a more powerful service, composite service. When conducting service composition, certain Quality of Service (*QoS*) constraints have to be considered, namely, *QoS*-aware Web service composition. This usually refers to the problem of composing a set of appropriate services into a richer service that follows application logics while satisfying certain *QoS* requirements.

QoS-aware Web service composition has been widely researched in the areas of Service Oriented Architecture (SOA) and Service Oriented Computing (SOC) [4,10,19]. However, existing approaches assume simple service composition models. Also, they give a single objective semi-optimal solution rather than a set of

Pareto optimal solutions that exhibit the trade-offs among different objectives. For example, it becomes complex if a client wants to make sure of receiving a service which meets a specific performance within a given cost level and a minimum time delay, but within a higher availability. This is because different dimensional qualities may conflict with one another in the real world. A typical example is the time and cost pair. QoS-aware service composition is then a multi-objective optimization problem, which requires simultaneous optimization of multiple and often competing criteria. Finding the optimal solutions for QoS-aware Web service composition with conflicting objectives and various restrictions on quality matrices is an NP-hard problem.

In the literature, linear weight sum method is employed, and single-objective algorithms are used to solve this problem [22]. However, linear weight sum method has the following problems: 1) solutions are sensitive to the weight vector and stronger prior awareness is required before solving the problem; 2) its number of solutions is small and the distribution of solutions is poor; 3) its time complexity increases exponentially with the increasing problem space size; 4) it will fail to find Pareto optimal solutions which lie in concave regions of the Pareto front.

On the other hand, linear weight sum method offers the user only one solution, while in reality, the user might prefer to see several good solutions, i.e., Pareto optimal, and decide which one is the best for himself. It is more natural to let the user decide the importance of each objective than aggregating the objectives and ask the user to specify a priori his/her preferences which is a demanding task. By using multi-objective optimization, it is no longer necessary for the user to define a priori an aggregation function.

Reinforcement learning (RL) [15] originally stems from the studies of animal intelligence, and has been developed as a major branch of machine learning for solving sequential decision-making problems. RL is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. RL has primarily been limited in its applicability to solve only single objective problems. However, many industrial and scientific problems are inherently complex and cannot be expressed in terms of just a single objective. Multi-objective Reinforcement Learning (MORL) combines advances in multi-objective optimization and techniques from reinforcement learning, thus extending RL techniques into the realms of multi-objective problems.

In this paper, an approach based on (MORL) is proposed for multi-objective service composition and adaptation in dynamic uncertain environments. Within the proposed approach, two algorithms are devised to handle different composition scenarios based on user preferences. Experiments have shown the ability of the proposed approach to provide scalable results especially in compositions with multiple quality attributes. The rest of this paper is organized as follows. The problem formulation and basic definitions are introduced in Section 2. Section 3 presents the multi-objective service composition approach. In Section 4, some experimental results are presented for evaluating the proposed approach.

Section 5 gives a brief review of related work and discussions. Finally, the paper is concluded in Section 6.

2 Problem Formulation

In this section, we describe the problem of service composition and give basic definitions related to our approach. In this approach, we employ the concept of Markov Decision Process (MDP) to schematically describe the process of service composition and adaptation. MDP is an AI method to model sequential decision processes under uncertainty and has also been used in different applications [12]. We use Multi-objective Markov Decision Process (MOMDP) to model multi-objective service composition in uncertain dynamic environments. The key concepts used in our approach are formally defined as follows.

In general, Web services can be described in terms of their service ID and *QoS*. A Web service can be formally defined by Definition 1.

Definition 1: (Web Service). A *Web Service WS* is defined as a tuple $WS =< ID, QoS >$, where ID is the identifier of the Web service, QoS is the quality of the service represented by a n-tuple $< Q_1; Q_2; ...; Q_n >$, where each Q_i denotes a QoS attribute of WS.

Generally, a single objective Markov Decision Process (MDP) can be defined defined as follows.

Definition 2: (Markov Decision Process (MDP)). An *MDP* is defined as a 4-tuple $MDP =< S, A, P, R >$, where

- S is a finite set of states of the world;
- $A(s)$ is a finite set of actions depending on the current state $s \in S$;
- P is a probability value, i.e., when an action $a \in A$ is performed, the world makes a probabilistic transition from its current state s to a resulting state s' according to a probability distribution $P(s' \mid s, a)$; and
- R is a reward function. Similarly, when action a is performed the world makes its transition from s to s' , the composition receives a real-valued reward r, whose expected value is $r = R(s' \mid s, a)$.

By extending the single-objective Markov decision process, the multi-objective Markov decision process is defined as follows.

Definition 3: (Multi-Objective Markov Decision Process (MOMDP)). An *MOMDP* is defined where

- There is an environment and an agent which takes an action at discrete time $t = 1, 2, 3,$.
- The agent receives a state $s \in S$ from the environment, where S is the finite set of states.
- The agent takes an action $a \in A$ at state s, where A is the finite set of actions that the agent can select.

- The environment gives the agent the next state $s' \in S$. The next state is determined with the state transition probability $P(s, a, s')$ for state s, action a and the next state s'. The state transition probability can be defined by the mapping:

$$P : S \times A \times S \to [0, 1] \tag{1}$$

- There are ($M > 1$) objectives which the agent wants to achieve, and the agent gains the following reward vector from the environment when it moves to the next state.

$$r(s, a, s') = [r_1(s, a, s'), r_2(s, a, s'), \cdots, r_M(s, a, s')]^T \tag{2}$$

MOMDP involves multiple actions and paths for each agent to choose. By using MOMDP to model service compositions, the composition agent will be able to find a set of Pareto optimal workflows satisfying the trade-offs among multiple QoS objectives. For each agent i, we call our service composition model as Multi-Objective Markov Decision Process based Web Service Composition ($MOMDP - WSC$), which simply replaces the actions in a MOMDP with Web services.

Definition 4: (MOMDP-Based Web Service Composition (MOMDP-WSC)). An *MOMDP-WSC* is defined as a 6-tuple $MOMDP - WSC =< S^i, s_0^i, S_r^i, A_i(.), P^i, R^i >$, where

- S^i is a finite set of world states observed by agent i;
- $s_0^i \in S$ is the initial state and any execution of the service composition usually starts from this state;
- $S_r^i \subset S$ is the set of terminal states. Upon arriving at one of those states, an execution of the service composition terminates;
- $A^i(s)$ is the set of Web services that can be executed in state $s \in S^i$, a Web service ws belongs to A^i, only if the precondition ws^P is satisfied by s;
- P^i is the probability when a Web service $ws \in A^i(s)$ is invoked when agent i makes a transition from its current state s to a resulting state s', where the effect of ws is satisfied. For each s, the transition occurs with a probability $P^i(s'|s, ws)$; and
- R^i is a reward function when a Web service $ws \in A^i(s)$ is invoked, agent i makes a transition from s to s', and the service consumer receives an immediate reward r^i, whose expected value is $R^i(s'|s, ws)$. Consider selecting Web service ws with multiple QoS criteria, agent i receives the following reward vector:

$$Q(s, ws, s') = [Q_1(s, ws, s'), Q_2(s, ws, s'), \cdots, Q_M(s, ws, s')]^T, \tag{3}$$

where each Q_i denotes a QoS attribute of ws.

The solution to an MOMDP-WSC is a decision policy, which is defined as a procedure for service selection $ws \in A$ by agent i in each state s. These policies, represented by π, are actually mappings from states to actions, defined as:

$$\pi : S \longrightarrow A. \tag{4}$$

Each policy of MOMDP-WSC can define a single workflow, and therefore, the task of our service composition model is to identify the set of Pareto optimal policies that gives the best trade-offs among multiple QoS criteria.

3 Multi-Objective Reinforcement Learning for Service Composition

In order to solve the above mentioned MOMDP, we propose an approach based on Multi-Objective Reinforcement Learning (MORL). The goal of MORL is to acquire the set of Pareto optimal policies in the MOMDP model. The set π^p of the Pareto optimal policies is defined by:

$$\pi^p = \left\{ \pi^p \in \Pi \middle| \nexists \pi \in \Pi, s.t. \vee^{\pi^p}(s) >_p \vee^{\pi}(s), \forall s \in S \right\}, \tag{5}$$

where Π is the set of all policies and $>_p$ is the dominance relation. For two vectors $a = (a_1, a_2, \cdots, a_n)$ and $b = (b_1, b_2, \cdots, b_n)$, $a >_p b$ means that $a_i \geq b_i$ is satisfied for all i and $a_i > b_i$ is satisfied for at least one i. Moreover, $V^\pi(s) = (V_1^\pi(s), V_2^\pi(s), \cdots, V_M^\pi(s))$ is the value vector of state s under policy π and it is defined by:

$$V^\pi(s) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s \right\}, \tag{6}$$

where E_π is the expected value provided that the agent follows policy π, s_t is the state at time t, r_t is the reward vector at t and γ is the discount rate parameter. We also define the Q-learning [20] vector by:

$$Q_\pi(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| s_t = s, a_t = a \right\}, \tag{7}$$

where a_t is the action at time t.

The MORL agent works to find the set of Pareto optimal policies under the condition that the agent does not know the state transition probability $P(s, a, s')$ and the expected reward vector $E\{r(s, a, s')\}$.

Current MORL approaches can be divided into two classes based on the number of policies that they learn [11]. The first class aims to learn the single policy that best satisfies a set of preferences between objectives as derived from the problem structure. We will refer to theses as single policy approaches. The second class seeks to find the set of policies which approximate the Pareto optimal front of all possible user preferences. We will refer to these as multiple policy approaches. Inspired by recent works in MORL [11], we propose two algorithms to address multi-objective composition in Web service environments. The first algorithm handles the case of single policy multi-objective service composition and the second algorithm handles the case of multiple policy multi-objective service composition.

3.1 Single Policy Multi-objective Service Composition

In the first algorithm, each QoS-objective is implemented as a separate Q-learning agent. Web services and their relative importance to these objectives are learned rather than predefined and the deployment of multiple QoS-objectives is enabled. At every state s, each agent i selects the candidate web service ws_i that optimizes its relative QoS-objective, then the agents negotiate together to decide which candidate service to execute in this state.

The agents learn to cooperate by negotiation and the agent that wins is the agent that would suffer the most if it did not. Given a state s, the agents suggest their Web service selections with strengths or weights $W_i(s)$. The agent with the largest W values is then allowed to deploy its preferred Web service in this state such that:

$$W_k(s) = Max_{i \in 1,\cdots,n} W_i(s) \tag{8}$$

Therefore, agent k is then a winner and executes Web service ws_k. We call agent k the leader in competition for state s at the moment. The agents then modify their $w_i(s)$ values based on whether they were obeyed, and what happened if they weren't, so the next round there may be a new winner.

Algorithm 1. Single Policy Algorithm

```
Observe state $s$
initialize leader $k$ with a random integer between 1 and $N$
$W_k \leftarrow 0$
$a_k \leftarrow argmax_a Q_k(s,a)$
repeat
  for all agents $i$ except $k$ do
    $W_i \leftarrow max_a Q_i(s,a) - Q_i(s,a_k)$
    if the highest $W_i > W_k$ then
      $W_k \leftarrow W_i$
      $a_k \leftarrow argmax_a Q_i(s,a)$
      $k \leftarrow i$
    end if
  end for
until converges
return $a_k$
```

W values build up on the difference between predicted reward P, which represents what is predicted if the agent was obeyed, and actual rewards A, which represents what actually happened. Therefore, W is calculated by:

$$W = P - A, \tag{9}$$

where p is the anticipated Q-vector if this agent's suggested Web service is executed, and A is the received Q-vector of the execution of another agent's suggested Web service. $(P - A)$ is the loss that the other agent causes to this one by being obeyed in its place. Consider the Q-learning process, when agent k is the winner and has its Web service executed, all other agents except k update their W values as follows:

$$W_i(x) \to (Q_i(x, a_i) - (r_i + \gamma max_{b \in a} Q_i(y, b))), \tag{10}$$

where the reward r_i and the next state y are caused by the agent k than by this agent itself. This process is described by Algorithm 1.

3.2 Multiple Policy Multi-objective Service Composition

In the second algorithm, the multiple policy service composition problem is solved by introducing the concept of the convex hull into Q-learning based Web service composition [8]. The convex hull is defined as the smallest convex set that contains all of a set of points. In this case, we mean the points that lie on the boundary of this convex set, which are of course the extreme points, the ones that are maximal in some direction. This is somewhat similar to the Pareto front, since both are maxima over trade-offs in linear domains. The proposed algorithm exploits the fact that the Pareto optimal set of the Q-vectors is the same as the convex hull of these Q-vectors.

In order to acquire the set of Pareto optimal service selection policies for all the *QoS* objectives, the set of the vertices in the convex hull of the Q-vectors at state s is updated by the value iteration method:

$$\hat{Q}(s, a) = (1 - \alpha)\hat{Q}(s, a) + \alpha \left[\boldsymbol{r}(s, a) + \gamma hull \bigcup_{a'} \hat{Q}(s', a') \right], \tag{11}$$

where $\hat{Q}(s, a)$ is the vertices of the convex hull of all possible Q-value vectors for taking action a at state s, α is the learning rate, γ is the discount value, r is the immediate reward, the operator hull means to extract the set of the vertices of the convex hull from the set of vectors.

Algorithm 2. Multiple Policy Algorithm

```
initialize $\hat{Q}(s, a)$ arbitrarily $\forall s, a$
while not converged do
  for all $s \in S, a \in A$ do
    $\hat{Q}(s, a) = (1 - \alpha)\hat{Q}(s, a) + \alpha \left[ \boldsymbol{r}(s, a) + \gamma hull \bigcup_{a'} \hat{Q}(s', a') \right]$
  end for
end while
```

Given these definitions, now we can rewrite the Q-learning based Web service composition algorithm [8] in terms of operations on the convex hull of Q-values. In the proposed algorithm, an action is selected based on the dominance relation between Q-vectors following the ϵ-greedy exploration strategy. This algorithm can be viewed as an extension to [8], where instead of repeatedly backing up maximal expected rewards, it backs up the set of expected rewards that are maximal for some set of linear preferences. The proposed multiple policy Web service composition algorithm is illustrated in Algorithm 2.

4 Simulation Results and Analysis

Two simulation experiments have been conducted to evaluate the proposed algorithms from different perspectives. The first experiment examines the ability of the single policy algorithm in composing Web services with Multiple QoS criteria and unknown user preferences. The second experiment examines the efficiency of the second algorithm in learning the set of Pareto optimal compositions considering the trade-offs among QoS objectives, simultaneously. Note that terms such as criteria and objectives, qualities and characteristics, solutions and workflows are used interchangeably unless otherwise specified.

We consider using four abstract services (i.e. the typical travel scenario) in both experiment. We assume there are a number of concrete Web services available for each abstract service. The detailed task is to choose the optimal concrete services to achieve better composition results that satisfy three QoS objectives which are *availability*, *response time* and *cost*.

4.1 Experiment Setting

Since there is not any sizable Web service test case that is in the public domain and that can be used for experimentation purposes, we focus on evaluating the proposed algorithms by using synthetic Web services. We assigned each concrete Web service in the simulated MOMDP-WSC model with random QoS vector. The values of the quality parameters in this vector followed normal distribution.

The proposed algorithms run in successive iterations/episodes till reaching a convergence point. Each algorithm converges to a near optimal policy once it receives the same approximate value of average accumulative rewards for a number of successive episodes, those average accumulated rewards are compared episode by episode and the difference is projected against a threshold. For both algorithms, this threshold value is set to 0.001, and the number of successive episodes is set to 1000

To ensure the highest learning efficiency, a number of parameters are set up for both experiments as follows. The learning rate α is set to 1, the discount factor γ is set to 0.8 and the ϵ-greedy exploration strategy value is set to 0.7. These parameter settings are shown in Table 1. The two experiments are conducted on 3.33 GHz Intel core 2 Duo PC with 3 GB of RAM.

Table 1. Parameter Settings

Parameter	Meaning	Value
α	Learning rate	1
γ	Discount factor	0.8
ϵ	Exploration strategy	0.7

4.2 Result Analysis

The results of the two experiments are demonstrated and analyzed in details in the following subsubsections

Experiment 1: Single Policy Algorithm
The purpose of the first experiment is to examine the ability of the single policy algorithm in composing web services with multiple QoS criteria and with no predefined user preferences. The algorithm's ability is measured in terms of the average accumulated reward the composition agent receives when it converges to an optimal policy. This reward value represents the aggregate QoS of the optimal workflow.

For this end, we ran the experiment multiple times and changed the environment scale in every run. The environment scale represents the number of concrete Web services assigned to each abstract service. The average accumulated reward of the single policy algorithm is recorded accordingly and compared with the average accumulated reward of the linear weight Q-learning approach [18]. The linear weight Q-learning approach assumes a predefined user preferences encoded as a weight vector over the multiple QoS attributes. This weight vector is set, in this experiment, to $\boldsymbol{\omega} = (0.3, 0.3, 0.3)$

Fig. 1 depicts the relationship between the average accumulated rewards obtained by running the single policy algorithm and the linear weight Q-learning approach multiple times with various number of concrete Web services.

As shown in Fig. 1, the proposed single policy algorithm yields higher rewards than the linear weight Q-learning approach, every run, apart from the number of concrete Web services. This proves the capability of the single policy algorithm to find higher quality compositions considering multiple QoS objectives. The reward difference becomes more significant as the number of web services increases, i.e., goes beyond 200. This is explained by the ability of the single policy algorithm to better explore the Pareto front. While the linear weight Q-learning approach fails to explore solutions lie on concave regions of the Pareto front, the proposed algorithm is able to scale well with the spread of Pareto front as the environment scale increases. Also, the linear-weight Q-learning approach assumes the usage of a predefined user preferences represented by a given weight vector $\boldsymbol{\omega}$. This weight vector might trip the search process into suboptimal regions of the Pareto surface as the composition agent is biased towards the user preferences. In contrast, the proposed algorithm builds upon the composition structure to derive the relative weights among different *QoS* preferences. This feature allows the proposed algorithm to adapt efficiently to the dynamics of open environments where many Web services join or leave during run-time.

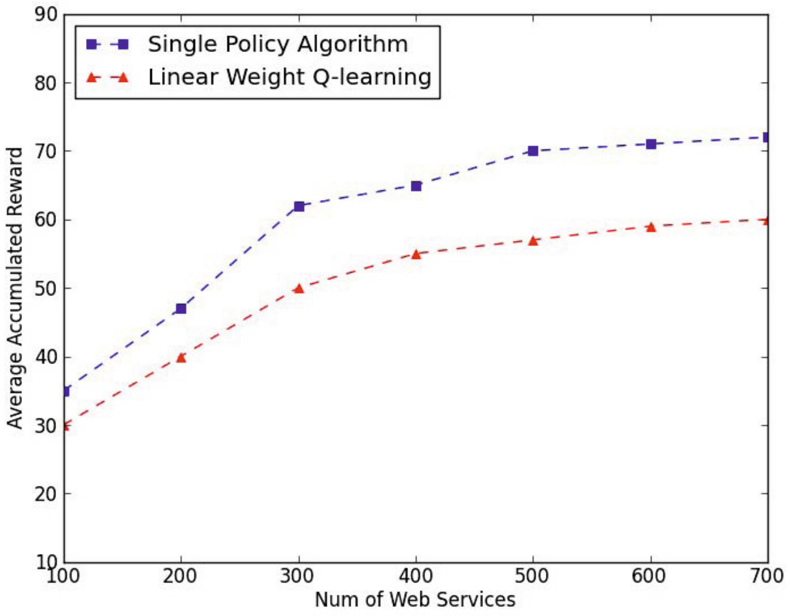

Fig. 1. Single Policy Algorithm

Experiment 2: Multiple Policy Algorithm

The purpose of the second experiment is to assess the ability of the proposed multiple policy algorithm in learning the set of Pareto optimal workflows considering the trade-offs among different *QoS* criteria. Totally three tests are carried out in this experiment. In the first two tests, each abstract service has been assigned 50 and 100 candidate Web services, respectively. Consequently, this creates an 4×50 matrix and 4×100 matrix for each quality attribute, respectively. The proposed multiple policy algorithm is implemented and tested with the parameters given above. The proposed algorithm runs till convergence and the number of non-dominated solutions/workflows are calculated accordingly.

As shown in Fig. 2, the experimental results indicate that the proposed algorithm is capable of guiding the search towards the Pareto-optimal front efficiently. As the initial attribute matrix data are created randomly, we have no idea where the true Pareto optimal front is. However, we understand that better solutions would be the ones with lower cost, lower response time, but higher availability. The search process should converge towards this direction.

Fig. 2a clearly shows that the optimal solutions have achieved lower cost and response time, but greater availability, which are centered between 0.4, 0.2, and 0.8, respectively. Fig. 2b also supports this statement, regardless of the bigger number of concrete services assigned to each abstract service, as the optimal solutions continue showing the same trend with lower cost and response time, but greater availability, which are centered between 0.3, 0.4, and 0.6, respectively.

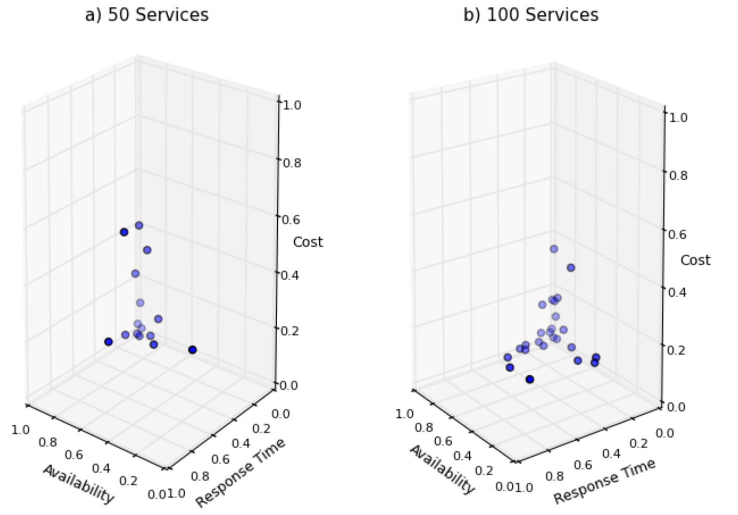

Fig. 2. (a) Results of composition with 50 services in each state; (b) Results of composition with 100 services in each state

The next test is performed to display the convergence property with the presence of different environment scales and various concrete services. Still, four abstract services are considered. We experiment three different cases with the number of concrete Web services varying from 100 to 400 for each abstract service. As shown in Fig. 3, it takes longer to find a set of optimal solutions with the increase of the number of concrete services. For example, in the case of 100 services, the algorithm converges at 400 episodes, while for the cases of 200 services and 400 services, the algorithm finds the non-dominated solutions at 800 episodes and 1000 episodes, respectively. The same tendency is anticipated to continue for any other bigger number of concrete services. As a matter of fact, the three cases generated the same number of non-dominated solutions, 25, at episode 400. The reason for this is currently unknown and is set for future research. In short, the proposed multiple policy algorithm is able to provide a set of Pareto-optimal solutions for service composition problems with different QoS criteria.

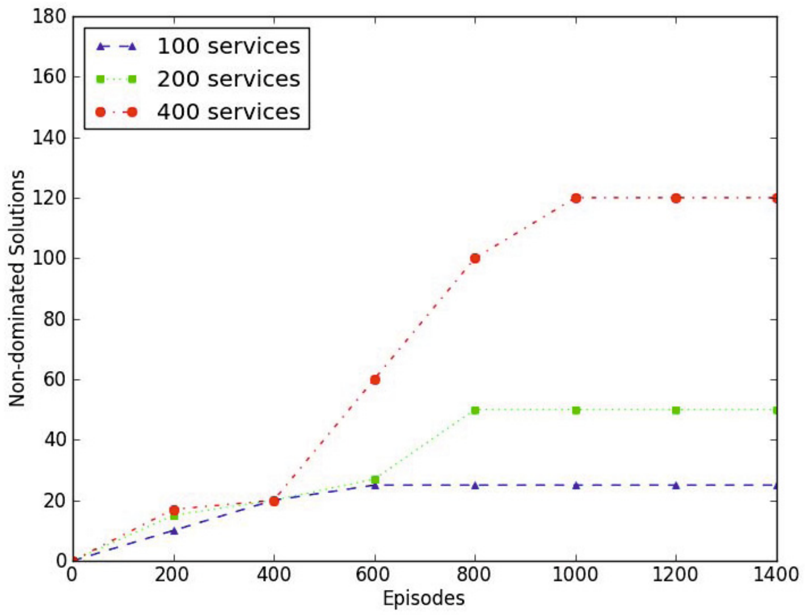

Fig. 3. Multiple Policy Algorithm

5 Related Work and Discussion

The problem of QoS-aware Web service composition is well known in SOC domain and various solutions are proposed based on different approaches[9,22,10,1]. Zeng et al. [22] introduced a QoS model in which aggregation functions are defined in order to aggregate and measure constraints into a single objective function. The major issues of the QoS-driven service selection approach presented in [22] are scaling (amongst objectives) and weighting. Its weighting phrase requires the selection of proper weights to characterize the users preferences, which can be very difficult in practice. Furthermore, the method from [22] cannot always guarantee the the fulfillment of global constraints, since Web service composition is not separable. Wang et al. [19] proposed an efficient and effective QoS-aware service selection approach. It employs cloud model to compute the QoS uncertainty for pruning redundant services while extracting reliable services. Then, Mixed Integer Programming (MIP) is used to select optimal services. Lin [10] aims at enhancing the credibility of service composition plan, taking advantage of a Web services QoS history records, rather than using the tentative QoS values advertised by the service provider, but at last the composition optimization problem is also instantiated into an Integer Programming (IP) problem. However, as pointed out by Berbner et al. in [1], the IP approach is hardly feasible in dynamic real-time scenarios when a large number of potential Web services are concerned. Canfora et al. [2] proposed the use of Genetic Algorithms (GAs) for the problem mentioned above. It has shown that GAs outperform integer programming used in [22] when a large number of services are available. Moreover, GAs are more flexible than the MIP since GAs allow the consideration of nonlinear composition rules. Apparently, traditional GAs have some inherent limitations in solving *QoS*-aware composition problems as the the selection of the weights of characteristics is required in order to aggregate multi-objectives into a single objective function in GAs.

All the above mentioned approaches, however, cannot solve Web service selection with multiple *QoS* objectives and multi-constrain. They all assume multiple criteria, no matter whether they are competing or not, can be combined into a single criterion to be optimized, according to some utility functions. When multiple quality criteria are considered, users are required to express their preference over different, and sometimes conflicting, quality attributes as numeric weights. This is a rather demanding task and an imprecise specification of the weights could miss user desired services.

Despite the fact that the *QoS* optimization problem is multi-objective by nature few approaches based on multi-objective algorithms can be found in the literature [17,6,16]. Yu and Lin [21] studied multiple *QoS* constraints. The composition problem is modelled as a Multi-dimension Multi-choice 0-1 Knapsack Problem (MMKP). A Multi-Constraint Optimal Path (MCOP) algorithm with heuristics is presented in [21]. However, the aggregation of parameters using the Min function is neglected. Maximilien and Singh [13] describe the Web Service Agent Framework (WSAF) to achieve service selection by considering the preferences of several service consumers as well as the trustworthiness of providers.

Evolutionary Algorithms (EAs) are suitable to solve multi-objective optimization problems because they are able to produce a set of solutions in parallel. A growing interest in the application of EAs to the multi-objective Web service composition in recent years is evident. Claro et al. [5] discussed the advantages of Multi-Objective Genetic Algorithms (MOGA) in Web service selection and a popular multi-objective algorithm, NSGA-II [7], is used to find optimal sets of Web services. Other EAs that have been proposed to solve multi-objective service composition include, Multi-Objective Particle Swarm Optimizer (MOPSO) [3], and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [14]. These EAs propose mathematical improvements to solve multi-objective service composition problems. However, as the dimensionality of problems increases, the performance of these EAs significantly deteriorates, since they cannot find a wide range of alternative solutions. In addition, MOGA and MOPSO cannot solve the optimization problems with concave Pareto fronts which are commonly encountered in the real world. In contrast, the proposed MORL based approach is able explore well the Pareto front of multi-objective service composition problems and deliver optimal solutions.

On the other hand, EAs require a level of awareness of the problem domain to setup the initial population through encoding the available combinations as genomes. In contrast, the proposed MORL based approach can learn how to best select Web services in complex environments based on multiple *QoS* criteria without any prior knowledge regarding the nature or the dynamics of these environment. Up to our knowledge, this is the first approach that uses MORL to solve this problem.

6 Conclusion

This paper proposes a novel approach to facilitate the QoS-aware service composition problem. By using multi-objective reinforcement learning, we devise two algorithms to enable Web service composition considering multiple QoS objectives. The first algorithm addresses the single policy composition scenarios, while the second algorithm addresses the multiple policy composition scenarios. The simulation results have shown the ability of the proposed approach to efficiently compose Web services based on multiple QoS objectives, especially in scenarios where no prior knowledge of QoS data is available and no predefined user preferences are given. The future work is set to study the performance of the proposed approach in large scale service compositions scenarios.

References

1. Berbner, R., Spahn, M., Repp, N., Heckmann, O., Steinmetz, R.: Heuristics for qos-aware web service composition. In: International Conference on Web Services, ICWS 2006, pp. 72–82 (2006)
2. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation GECCO 2005, pp. 1069–1075. ACM, New York (2005)
3. Cao, J., Sun, X., Zheng, X., Liu, B., Mao, B.: Efficient multi-objective services selection algorithm based on particle swarm optimization. In: 2010 IEEE Asia-Pacific Services Computing Conference (APSCC), pp. 603–608 (2010)
4. Chiu, D., Agrawal, G.: Cost and accuracy aware scientific workflow composition for service-oriented environments. IEEE Trans. Services Computing (2012)
5. Claro, D.B., Albers, P., Hao, J.K.: Selecting web services for optimal composition. In: SDWP 2005, pp. 32–45 (2005)
6. de Campos, A., Pozo, A.T.R., Vergilio, S.R., Savegnago, T.: Many-objective evolutionary algorithms in the composition of web services. In: 2010 Eleventh Brazilian Symposium on Neural Networks (SBRN), pp. 152–157 (2010)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. Dehousse, S., Faulkner, S., Herssens, C., Jureta, I.J., Saerens, M.: Learning optimal web service selections in dynamic environments when many quality-of-service criteria matter. Machine Learning, InTech., 207–229 (2009)
9. Kalasapur, S., Kumar, M., Shirazi, B.A.: Dynamic service composition in pervasive computing. IEEE Trans. Parallel and Distributed Systems 18(7), 907–918 (2007)
10. Lin, W., Dou, W., Luo, X., Chen, J.: A history record-based service optimization method for qos-aware service composition. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 666–673 (2011)

11. Liu, C., Xu, X., Hu, D.: Multiobjective reinforcement learning: A comprehensive overview. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews PP(99), 1–13 (2013)
12. Mastronarde, N., Kanoun, K., Atienza, D., Frossard, P., van der Schaar, M.: Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems. IEEE Trans. Multimedia 15(2), 268–278 (2013)
13. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic web services selection. IEEE Internet Computing 8(5), 84–93 (2004)
14. Suciu, M., Pallez, D., Cremene, M., Dumitrescu, D.: Adaptive moea/d for qos-based web service composition. In: Middendorf, M., Blum, C. (eds.) EvoCOP 2013. LNCS, vol. 7832, pp. 73–84. Springer, Heidelberg (2013)
15. Sutton, R.S., Barto, A.G.: Reinforcement learning: Introduction (1998)
16. Taboada, H.A., Espiritu, J.F., Coit, D.W.: Moms-ga: A multi-objective multi-state genetic algorithm for system reliability optimization design problems. IEEE Transactions on Reliability 57(1), 182–191 (2008)
17. Wada, H., Suzuki, J., Yamano, Y., Oba, K.: E3: A multiobjective optimization framework for sla-aware service composition. IEEE Transactions on Services Computing 5(3), 358–372 (2012)
18. Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A.: Adaptive service composition based on reinforcement learning. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 92–107. Springer, Heidelberg (2010)
19. Wang, S., Zheng, Z., Sun, Q., Zou, H., Yang, F.: Cloud model for service selection. In: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 666–671 (2011)
20. Watkins, C.: Learning from Delayed Rewards. PhD thesis, Cambridge University, England (1989)
21. Yu, T., Lin, K.-J.: Service selection algorithms for composing complex services with multiple qos constraints. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 130–143. Springer, Heidelberg (2005)
22. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)