Multi-Objective Service Composition in Uncertain Environments

Ahmed Mostafa and Minjie Zhang, Senior Member, IEEE

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 service composition techniques are simply based on the assumption that multiple QoS 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 these utility functions or weights are not well-known a priori. In addition, the existing approaches are designed to work in certain environments, where the QoS parameters are well-known in advance. These approaches will render fruitless when facing uncertain and dynamic environments, e.g., cloud environments, where no prior knowledge of the QoS parameters is available. In this paper, two novel multi-objective approaches are proposed to handle QoS-aware Web service composition with conflicting objectives and various restrictions on the quality matrices. The proposed approaches use reinforcement learning in order to deal with the uncertainty characteristics inherent in open and dynamic environments. Experimental results reveal the ability of the proposed approaches to find a set of Pareto optimal solutions, which have the equivalent quality to satisfy multiple QoS-objectives with different user preferences.

Index Terms—web services; multi-objective composition; uncertain environments;

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, i.e., a 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 the application logic 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) [7], [10], [20]. However, most of the existing approaches for service composition are developed based on a single objective semi-optimal solution, rather than a set of Pareto optimal solutions that exhibit the trade-offs among different QoS objectives. For example, it becomes complex if a consumer wants to make sure of receiving a service that 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 of conflicting qualities 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 the quality matrices is then an NP-hard problem.

In the literature, a linear weight sum method is employed, and single-objective algorithms are used to solve this problem [25]. However, the linear weight sum method holds the following limitations: 1) its 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 might fail to find Pareto optimal solutions, if these solutions lie in concave regions of the Pareto front; and 5) it offers users only one solution, while in reality, users might prefer to see several good solutions, i.e., Pareto optimal, and decide which one is the best for their preferences.

Meanwhile, dealing with dynamicity and uncertainty features inherent in cloud environments has been a standing challenge. Most of the current multi-objective service composition approaches carry the following drawbacks which are that: 1) they adopt mathematical optimization techniques which are built to handle static environments, where the QoS parameters are certain and well-known in advance; 2) they assume the existence of an explicit model of the service environment before starting the composition process; and 3) they only consider static environments where the relative weight among multiple quality
objectives is fixed, while in dynamic environments, this relative weight might change automatically.

Clearly, these approaches might fail when dealing with cloud environments where 1) the QoS parameters are unknown; 2) service providers can join or leave the environment during runtime; and 3) there are no central registries to record the available service providers and their corresponding QoS parameters. For these reasons, service composition in cloud environments needs to be learned than optimized, where the service composition agent ought first to interact with the cloud network, discover available service providers and assert their quality parameters, before making contracting decisions.

Reinforcement learning (RL) is a powerful technique, for solving sequential decision-making problems, that has developed as a major branch of machine learning in the past few years [15]. RL is concerned with how an artificial agent ought to take actions in uncertain environments, in order to maximize some notions of long-term rewards. 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, two approaches based on MORL are proposed in order to address multi-objective service composition in uncertain and dynamic environments. Towards this end, the proposed approaches adopt an online learning scheme where the learning agent continuously explores the service environment in a sequence of episodes. Therefore, the proposed approaches can adapt to the situations where multiple providers join or leave the service environment during runtime. The experiments have shown the ability of the proposed approaches to provide scalable results, especially in service compositions with multiple quality attributes. The contributions of this paper are as follows.

- First, a new method is proposed to handle multi-objective service composition in uncertain and dynamic environments, using reinforcement learning.
- Second, a novel approach is devised for single policy multi-objective service composition. This approach adopts a self-organization mechanism that exploits the problem structure, in order to derive the weights of different QoS objectives. Hence, the users need not to explicitly assign a weight for each QoS objective.
- Third, an innovative approach is proposed for multiple policy multi-objective service composition. This approach leverages the convex hull operator to simultaneously find the set of optimal solutions, which satisfy all the trade-offs among different QoS parameters.
- Fourth, the efficiency and effectiveness of the proposed approaches are evaluated through a comprehensive set of experiments.

The rest of this paper is organized as follows. A motivating scenario is introduced in Section 2. The problem formulation and the basic service composition model are introduced in Section 3. Section 4 presents the multi-objective service composition approaches. In Section 5, extensive experimental results are presented for evaluating the proposed approaches. Section 6 gives a brief review of the related work. Discussions of the proposed approaches are presented in Section 7. Finally, the paper is concluded in Section 8.

2 A Motivating Scenario

In this section, a big data service provisioning scenario is presented in order to motivate the challenge of multi-objective service composition in uncertain and dynamic environments. Let us consider Company A that accumulates terabytes of transactional customer data and needs to analyze this data to gain meaningful insights to help customize its marketing campaigns. It becomes difficult for Company A to process this large amount of data using in-house processing tools. The cloud computing platform would then represent a convenient and cost-effective option for Company A to outsource this task. Company A decides to submit its raw transactional data to the analytic cloud service, e.g., Amazon Elastic Map Reduce (EMR). The analytic cloud service, e.g., Service S, will in turn generate a composite data analysis job on behalf of Company A. This data analysis job, in particular, has a number of tasks/abstract services, i.e., Software as a Service (SaaS), namely, the data cleaning service, (Task 1), the data transformation service, (Task 2), the pattern analysis services, (Task 3), and the pattern evaluation and presentation service, (Task 4). Besides these software services, the data analysis job also needs CPU, network and storage resources from Infrastructure as a Service (IaaS) providers, as shown in Fig. 1.

![Motivating Scenario](image)

Typically, there could be multiple service providers available to attain each of the above tasks/abstract services. In addition, these service providers usually compete with each other, offering various QoS parameters such as availability, cost and response time.
Therefore, Service S has to search the cloud service network to select a set of service providers that satisfy the customer requirements. In addition, Service S has to consider the trade-offs among conflicting QoS parameters in this selection. Due to the uncertain and dynamic nature of cloud environments, Service S does not know, in advance, the QoS parameters of these service providers. In addition, multiple service providers may join or leave the cloud network during runtime. As a result, the selection process would then become very challenging.

3 Problem Formulation

This section describes the problem of service composition and gives the basic definitions related to the proposed service composition model. In this model, the concept of Partially Observable Markov Decision Process (POMDP) in employed to schematically describe the process of service composition and adaptation. POMDP is an AI method that has been devised to model sequential decision processes under uncertainty, and has also been deployed in various applications [11]. We use Multi-Objective Partially Observable Markov Decision Process (MO-POMDP) to model multi-objective service composition in uncertain and dynamic environments. The key concepts used in the service composition model are formally defined as follows.

Basically, 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 < Q1, Q2, ..., Qn >, where each Qi denotes a QoS attribute of WS.

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

Definition 2: (Partially Observable Markov Decision Process (POMDP)). A POMDP is defined as a 6-tuple POMDP =< S, A, P, R, O, Z >, 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 ∈ S;
- P is a probability value, i.e., when an action a ∈ 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' | s, a);
- R is a reward function. Similarly, when action a is performed the world makes its transition from s to s', the decision agent receives a real-valued reward r, whose expected value is r = R(s' | s, a);
- O is the set of observations that the agent can receive about the world; and
- Z is the observation probability, the probability that the agent receives observation o', when making transition from s to s' by selecting action a, p(o' | s', a).

By extending the single objective POMDP, the multi-objective POMDP is defined as follows.

Definition 3: (Multi-Objective Partially Observable Markov Decision Process (MO-POMDP)). A MO-POMDP is defined where

- There is an environment where an agent takes an action at discrete time t = 1, 2, 3, ..., etc;
- The agent has received a state s ∈ S from the environment, where S is the finite set of states;
- The agent takes an action a ∈ 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' ∈ 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:

\[ P : S \times A \times S \rightarrow [0, 1] \]  (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.

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

MO-POMDP involves multiple actions and paths for each agent to choose. By using MO-POMDP to model service compositions in uncertain and dynamic environments, 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 Partially Observable Markov Decision Process based Web Service Composition (MO-POMDP-WSC), which simply replaces the states in a MO-POMDP with tasks, the actions with Web services, and the rewards with QoS vectors. For the rest of this paper, these terms will be used interchangeably unless otherwise specified.

Definition 4: (MO-POMDP-Based Web Service Composition (MO-POMDP-WSC)). A MO-POMDP-WSC is defined as a 8-tuple MO-POMDP-WSC =< S, s₀, S₁, A₁, P₁, R₁, O₁, Z₁ >, where

- S is the set of tasks in the state space of a particular workflow partially observed by agent i;
- s₀ ∈ S is the initial task and any execution of the workflow usually starts from this task;
- S₁ ⊂ S is the set of terminal tasks. Upon arriving at one of those tasks, an execution of the workflow terminates;
• \( A^i(s) \) is the set of Web services that can be executed in task \( s \in S^i \), Web service \( ws \) belongs to \( A^i \), only if the precondition \( ws^p \) is satisfied by \( s^i \);  
• \( P^i \) is the probability when Web service \( ws \in A^i(s) \) is invoked, then agent \( i \) makes a transition from its current task \( s \) to a resulting task \( s' \), where the effect of \( ws \) is satisfied. For each \( s \), a transition occurs with a probability \( P^i(s'|s,ws) \);  
• \( R^i \) is a reward function when 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 QoS vector as a reward:

\[
Q(s,ws,s') = [Q_1(s,ws,s'), Q_2(s,ws,s'), \ldots, Q_M(s,ws,s')]^T, (3)
\]

where each \( Q_i \) denotes a QoS attribute of \( ws \);  
• \( O^i \) is the set of observations received by agent \( i \), such that, \( o' \) is the observation that agent \( i \) receives when making a transition from unobserved task \( s \) to unobserved task \( s' \) by selecting service \( ws \). This observation gives the agent some evidence about the current task \( s \); and  
• \( Z^i \) is the observation probability. Since \( s \) is not known exactly, \( Z^i \) defines a probability distribution of observations over available tasks \( p(o'|s',ws) \).

The solution to an MO-POMDP-WSC is a set of decision policies. Each policy is defined as a procedure for service selection \( ws \in A \), by agent \( i \), for each task \( s \). These policies, represented by \( \pi \), are actually mappings from tasks to Web services, defined as:

\[
\pi : S \rightarrow A. (4)
\]

Each policy of a MO-POMDP-WSC can define a single workflow, and therefore, the aim of our service composition model is to identify the set of Pareto optimal policies/workflows that can give the best trade-offs among multiple QoS criteria.

### 4 Multi-Objective Service Composition

In this section, we advocate the usage of Multi-Objective Reinforcement Learning (MORL), in order to solve the above mentioned MO-POMDP-WSC. The goal of MORL is to acquire the set of Pareto optimal policies in the MO-POMDP model. The set \( \pi^p \) of the Pareto optimal policies is defined by:

\[
\pi^p = \left\{ \pi^p \in \Pi \mid \exists \pi^p \in \Pi, s.t. \forall \pi^p (s) >_p \forall \pi (s), \forall s \in S \right\}, (5)
\]

where \( \Pi \) is the set of all policies and \(>_p \) is the dominance relation.

#### Definition 5: (Dominance Relation)

For two vectors \( a = (a_1,a_2,\ldots,a_n) \) and \( b = (b_1,b_2,\ldots,b_n) \), \( a >_p b \) 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, \( V^\pi(s) = (V^\pi_1(s), V^\pi_2(s), \ldots, V^\pi_M(s)) \) is the value vector of state \( s \), under policy \( \pi \), and is defined by:

\[
V^\pi(s) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s \right\}, (6)
\]

where \( \mathbb{E}_\pi \) is the expected value provided that the agent follows policy \( \pi \), \( s_t \) is the state at time \( t \), \( r_t \) is the reward vector at \( t \) and \( \gamma \) is the discount rate parameter. The state-action value \( [21] \) for the pair \((s,a)\), under policy \( \pi \), can also be defined as follows:

\[
Q^\pi(s,a) = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}, (7)
\]

where \( a_t \) is the action at time \( t \). In the following subsections, a set of algorithms based on MORL are proposed to facilitate multi-objective service composition in uncertain and dynamic environments. For clarity, a summary of the most common notations, used throughout these algorithms, have been summarized into Table 1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( ws )</td>
<td>Web service</td>
</tr>
<tr>
<td>( s )</td>
<td>Finite set of states, i.e., tasks</td>
</tr>
<tr>
<td>( a )</td>
<td>Finite set of actions, i.e., Web services</td>
</tr>
<tr>
<td>( r )</td>
<td>Reward received when selecting a specific Web service, i.e., QoS</td>
</tr>
<tr>
<td>( \pi )</td>
<td>Selection policy</td>
</tr>
<tr>
<td>( \pi^p )</td>
<td>Pareto optimal policy</td>
</tr>
<tr>
<td>( Q(s,a) )</td>
<td>State-action value function</td>
</tr>
<tr>
<td>( P^\pi )</td>
<td>Predicted reward</td>
</tr>
<tr>
<td>( A )</td>
<td>Actual reward</td>
</tr>
<tr>
<td>( CH(S) )</td>
<td>Convex hull of a set of services ( S )</td>
</tr>
<tr>
<td>( Q(s,a) )</td>
<td>The convex hull of the ( Q )-value vectors at state ( s )</td>
</tr>
</tbody>
</table>

#### 4.1 Multi-Objective Reinforcement Learning for Service Composition

This subsection proposes a baseline algorithm for service composition using MORL. In this algorithm, the learning agent works to find the set of Pareto optimal policies/workflows under the condition that the agent does not know the environment model, i.e., state transition probability \( P(s,a,s') \) and the expected reward vector \( E(r(s,a,s')) \). Towards this end, the learning agent iterates over state-action pairs. Then, the agent updates its current hypothesis about the
actual state-action value, i.e., $Q(s,a)$, according to the following update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s,a)], \quad (8)$$

where $s$ represents the state space for the group of tasks that the agent needs to combine to achieve a user’s request, $a$ is the action vector representing the available Web services, $r$ is the reward given by selecting particular service, $\alpha$ is a learning rate which controls convergence and $\gamma$ is the discount factor that reflects the learning policy. At the right side of Equation 8, $\alpha [r + \gamma \max_{a'} Q(s', a') - Q(s,a)]$ represents the new immediate reward gained by assigning any Web service that belongs to $a$ to any task belongs to $s$; whereas $Q(s,a)$ is the previously observed reward.

Using the above paradigm, we now present a baseline MORL algorithm for service composition. This baseline algorithm is described by Algorithm 1.

**Algorithm 1 Baseline Multi-Objective Reinforcement Learning for Service Composition**

1: initialize $Q(s,a,o)$ arbitrarily
2: for each episode do
3: initialize state $s$
4: for each step of episode do
5: Choose action $a$ from $s$ (e.g. $\epsilon$-greedy )
6: observe state $s' \in S$ and reward vector $r^f \in \mathbb{R}$
7: for each objective $o \in M$ do
8: $Q(s,a,o) \leftarrow (1-\alpha)Q(s,a,o) + \alpha \left[ r^f(s,a,o) + \gamma \max_{a'} Q(s',a',o) - Q(s',a,o) \right]$
9: end for
10: $s \leftarrow s'$
11: end for
12: end for

Algorithm 1 works as follows. The learning agent starts by initializing the $Q$-values for each triple of states, actions and objectives (Line 1). Then, at each episode, the agent begins in State $s$, i.e., Task $t$ (Line 3), and selects an action $a$, i.e., Web service $w$, based on the $\epsilon$-greedy strategy as follows. For $\epsilon$ episodes, the learning agent selects the Web service with the highest QoS values. For $(1 - \epsilon)$ episodes, the agent selects a random Web service (Line 5). Upon taking action $a$, the agent is being transitioned into the new state $s'$ and the environment provides it with the vector of rewards $r^f \in \mathbb{R}$. The $Q(s,a,o)$ are updated with a multi-objective version of Equation 8 (Line 8). This process is repeated until the $Q$-values converge. The optimal policy then becomes the policy that is used to select the optimal action, i.e., the Web service with the highest quality parameters, at each state.

In order to learn the optimal selection policy for multi-objective service composition, we propose two approaches, in the following two subsections, respectively. The first approach introduces a self-organization mechanism that exploits the problem structure to learn the single policy that best satisfies a set of preferences among multiple QoS objectives. This approach is referred to as the single policy approach. The second approach uses the convex hull operator to simultaneously find the set of policies which approximate the Pareto optimal front of all possible user preferences. This approach is referred to as the multiple policy approach.

### 4.2 Single Policy Multi-Objective Service Composition

In this subsection, we propose a multi-agent self-organization mechanism to approach the challenge of multi-objective service composition. The proposed mechanism works as follows. Each QoS-objective is implemented as a separate 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. This self-organization mechanism is described by Algorithm 2.

**Algorithm 2 Single Policy Multi-Objective Service Composition**

1: Observe state $s$
2: initialize leader $k$ with a random integer between 1 and $N$
3: $Re_k \leftarrow 0$
4: $a_k \leftarrow \text{argmax}_a Q_k(s,a)$
5: repeat
6: for all agents $i$ except $k$ do
7: $Re_i \leftarrow \max_a Q_i(s,a) - Q_k(s,a)$
8: if the highest $Re_i > Re_k$ then
9: $Re_k \leftarrow Re_i$
10: $a_k \leftarrow \text{argmax}_a Q_i(s,a)$
11: $k \leftarrow i$
12: end if
13: end for
14: until converges
15: return $a_k$

At every state $s$, each agent $i$ selects the candidate web service $w_{s_i}$ that optimizes its own utility, i.e., maximize or minimize its relative QoS-objective, then the agents self-organize in order to decide which candidate service to execute in this state (Lines 1-2).

The agents learn to self-organize by adopting a regret minimization strategy, in which the agent that executes its candidate service is the agent that would suffer the most if it does not. Given a state $s$, the agents suggest their Web service selections with strengths/regrets $Re_i(s)$ (Lines 3-7). The agent, e.g., agent $k$, with the largest $Re$ values is then allowed to deploy its preferred Web service in this state such that:
Therefore, agent $k$ is then a leader and executes Web service $ws_k$ (Lines 8-12). We call agent $k$ the leader in this round for state $s$ at the moment. The agents then modify their $R_{e_i}(s)$ values based on whether their candidate services were executed, and what happened if these services were not executed, so there may be a new leader in the next round.

Regret values, i.e., $Re$, are built 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, $Re$ is calculated as follows:

$$Re = P - A,$$  \hspace{1cm} (10)

where $p$ is the anticipated QoS, if this agent’s suggested Web service is executed, and $A$ is the received QoS, as a result of the execution of another agent’s suggested Web service. $(P - A)$ is the loss that the other agent causes to this agent by being obeyed in its place. Consider the reinforcement learning process, i.e., Eq. 8, when agent $k$ is the leader and has its Web service executed, every other agent, except $k$, updates its regret value as follows:

$$Re_i(x) \rightarrow (Q_i(x, a_i) - (r_i + \gamma \max_{b \in Q_i(y,b)} Q_i(y,b))),$$  \hspace{1cm} (11)

where the reward $r_i$ and the next state $y$ are caused by the agent $k$, than by this agent itself.

### 4.3 Multiple Policy Multi-Objective Service Composition

In this subsection, we approach the multi-objective service composition problem by introducing the convex hull operator, a geometric operator for calculating the service selection policies from a set of services with multiple QoS attributes.

Convex hull \cite{4} is a well known geometric object that has been utilized in various applications \cite{18}. Generally, Convex hull is defined on a set of points in $d$-dimensional ($d > 1$) space as follows.

**Definition 6: (Convex Hull (CH)):** The convex hull is the smallest convex polygon that encloses all points of a set. Let $S = \{s_1, s_2, ..., s_n\}$ be a set of $n$ services with $d$ dimensional QoS vectors, $Q = \{Q_1, Q_2, ..., Q_m \in S\}$. The convex hull of $S$, denoted by $CH(S)$, is a sequence of services like $CH(S) = (s_{i_1}, s_{i_2}, ..., s_{i_m})$, where $m \leq n$ and $s_{i_j} \in S$ for $j = 1, 2, ..., m$. Such that, it covers all the services of $S$.

In this approach, we aim to identify the extreme services that lie on the boundaries of this convex set. Those extreme services could be the maximum or the minimum in any direction according to the given QoS objectives, e.g., upper right or lower left.

For example, consider having a set of services with two QoS objectives, e.g., availability and reputation. There could only be four-case optimizations (Min, Min), (Min, Max), (Max, Min), (Max, Max) (see Fig. 2). Therefore, the extreme services on the convex hull can be divided into four clusters. The first cluster (all services between $c$ and $d$ as in Fig. 2) contains all the services that are the best with respect to both objectives (assuming a maximization problem). The second and the third clusters contain the middle services, which are good in only one objective (all points between $b$ and $c$, and between $a$ and $d$, as in Fig. 2). The services of the final cluster (all services between $a$ and $b$ as in Fig. 2) are the worst services, with respect to both objectives.

This is somewhat similar to the concept of Pareto front, since both the Pareto front and the convex hull are optimal over trade-offs in linear domains. Therefore, the proposed approach exploits the fact that the Pareto front of a set of services is the same as the convex hull of these services.

One fast algorithm for computing the convex hull is the Quick Hull \cite{2}. This algorithm can be efficiently used for finding each quarter part of a convex hull. In the proposed approach, we employ the Quick Hull only for finding the quarter part of the convex hull that is relevant to the assigned objective function, e.g., the quarter between $c$ and $d$ in Figure 5. It is clear that each point on this quarter part of the convex hull is a non-dominated solution. This is the main idea in the relation between convex hulls and non-dominated solutions.

In the following, we propose a new approach based on the convex hull operator to obtain the set of optimal service selection policies for all QoS objectives, simultaneously. The proposed approach is described by Algorithm 3. In this approach, the multiple policy service composition problem is solved by introducing the convex hull operator into the MORL based Web service composition (refer to Algorithm 1).

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:

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

(12)

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

**Algorithm 3 Multiple Policy Multi-Objective Service Composition**

1: initialize \( \hat{Q}(s, a) \) arbitrarily \( \forall s, a 
2: for \text{episode } t = T \text{ downto } 1 \text{ do}
3: initialize state \( s 
4: \text{repeat}
5: Choose action \( a \) from \( s \) (e.g., \( \epsilon \)-greedy )
6: Take action \( a \) and observe next state \( s' \in S 
7: \text{Observe reward vector } \bar{r} \in \hat{R}
8: \max_{a'} \hat{Q}(s, a') \leftarrow \bigcup_{a'} \hat{Q}(s', a')
9: \hat{Q}(s, a) \leftarrow (1 - \alpha)\hat{Q}(s, a) + \alpha \left[ \bar{r}(s, a) + 
\gamma \text{hull} \bigcup_{a'} \hat{Q}(s', a') \right]
10: \text{ } s \leftarrow s'
11: until s is terminal
12: end for

Given the aforementioned concepts, now we can rewrite the MORL based Web service composition algorithm, i.e., Algorithm 1, in terms of operations on the convex hull of the Q-vectors. In the proposed algorithm, i.e., Algorithm 3, an action, i.e., Web service, is selected, based on the dominance relation between the Q-vectors, following the \( \epsilon \)-greedy exploration strategy (Lines 1-6). Then, instead of repeatedly backing up the maximal expected rewards, i.e., as in the single objective case [13], it backs up the set of expected rewards that are maximal for some set of linear preferences (Lines 7-10).

An important advantage of using the convex hull operator is that it takes into account both goals of multi-objective service selection: convergence to the Pareto-optimal front and diversity, simultaneously. This means that the proposed approach, without using any niching technique, or crowding distance [9], is able to find a good set of diverse solutions.

### 5 Simulation Results and Analysis

Four simulation experiments have been conducted to evaluate the proposed approaches from different perspectives. The first experiment examines the ability of the single policy multi-objective approach in exploiting the problem structure to compose Web services with multiple QoS criteria and unknown user preferences. The second experiment investigates the scalability of the single policy multi-objective approach in the face of service environments with various sizes. The third experiment examines the efficacy of the multiple policy multi-objective approach in learning the set of Pareto optimal compositions considering all the trade-offs among QoS objectives, simultaneously. The fourth experiment conducts a rigorous analysis and benchmarks the quality of the multiple policy multi-objective approach results with the state-of-the-art approaches in multi-objective service composition. Note that terms such as criteria and objectives, qualities and characteristics, solutions and workflows are used interchangeably unless otherwise specified.

In order to satisfy the user requirements, we consider an abstract workflow that consists of a number of tasks. In addition, we assume there are a number of concrete Web services that can match each task. The mission of a learning agent is to select the optimal concrete Web services for each task in order to achieve better composition results that satisfy three QoS objectives which are availability, response time and cost.

#### 5.1 Experiment Setting

For experimentation purposes, we focus on evaluating the proposed approaches by using synthetic Web services. Towards this end, each concrete Web service in the simulated MO-POMDP-WSC model is assigned with a random QoS vector. The values of the quality parameters in this vector are generated following the normal distribution.

In order to address uncertain service environments, the proposed approaches run in successive iterations/episodes till reaching a convergence point. Each approach converges to a near optimal policy once it receives the same approximate value of average accumulated rewards for a number of successive episodes. In order to simulate dynamic service environments where multiple service providers may join or leave during runtime, a certain number of services are designated to be replaced periodically. Towards this end, the QoS values of 10% of the Web services are varied periodically every 100 episodes.

The average accumulated reward for each workflow is computed by aggregating the QoS vectors of its member Web services using Table 2 as follows:

<table>
<thead>
<tr>
<th>QoS parameter</th>
<th>Aggregation function</th>
</tr>
</thead>
<tbody>
<tr>
<td>availability</td>
<td>( \sum_{i=1}^{n} \log(\text{availability}(ws_i)) )</td>
</tr>
<tr>
<td>response time</td>
<td>( \sum_{i=1}^{n} (\text{response time}(ws_i)) )</td>
</tr>
<tr>
<td>cost</td>
<td>( \sum_{i=1}^{n} \text{cost}(ws_i) )</td>
</tr>
</tbody>
</table>

These average accumulated rewards are compared episode by episode and the difference is projected against a threshold. This threshold can take any value...
according to the user preferred policy. To yield the highest quality solutions, tight convergence settings are adopted for both approaches. Therefore the threshold value is set to 0.001, and the number of successive episodes is set to 1000.

A number of parameters are set up for usage throughout the experiments to ensure the maximum learning efficiency. The values of these learning parameters are decided based on previous empirical simulations conducted by the authors [13] as follows. The learning rate $\alpha$ is set to 1, the discount factor $\gamma$ is set to 0.8 and the $\epsilon$-greedy exploration strategy value is set to 0.7. These parameter settings are shown in Table 3. All the experiments are conducted on 3.33 GHz Intel core 2 Duo PC with 3 GB of RAM.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Learning rate</td>
<td>1</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Discount factor</td>
<td>0.8</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>Exploration strategy</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### 5.2 Result Analysis

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

#### 5.2.1 Experiment 1: Evaluation of Single Policy Multi-Objective Composition

The purpose of Experiment 1 is to examine the ability of the single policy multi-objective approach in exploiting the problem structure to find high quality web services’ compositions with multiple QoS criteria and without predefined user preferences. The approach’s ability is measured in terms of the average accumulated reward the composition agent receives when the solution 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 task. The average accumulated reward of the single policy multi-objective approach is recorded accordingly and compared with the average accumulated reward of the linear weight Q-learning approach [19]. The linear weight Q-learning approach, however, assumes the usage of fixed user preferences represented by a given weight vector $\vec{\omega}$. This weight vector might trip the search process into suboptimal regions of the non-dominated service space as the composition agent is biased towards the user preferences. In contrast, the proposed single policy multi-objective approach builds upon the composition structure to derive the relative weights among different QoS preferences. This feature allows the proposed approach to adapt efficiently to the dynamics of open Web environments where the relative weight of each QoS preference is automatically changed by the change in the dynamic environment.

#### 5.2.2 Experiment 2: Scalability of Single Policy Multi-Objective Composition

The purpose of Experiment 2 is to examine the performance of the single policy multi-objective approach under different scales of service environments. Totally two tests are carried out in this experiment. In Test 1, the service environment scale is represented by the number of concrete services assigned to each approach yields higher rewards than that of the linear weight Q-learning approach, in every run, apart from the number of concrete Web services deployed. This proves the capability of the single policy multi-objective approach to find higher quality compositions considering multiple QoS criteria. The reward difference even 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 multi-objective approach to better explore the non-dominated services set. While the linear weight Q-learning approach fails to explore the services which lie on concave regions of the non-dominated service set, the proposed approach is able to scale well with the spread of this set as the environment scale increases. Also, the linear-weight Q-learning approach assumes the usage of fixed user preferences represented by a given weight vector $\vec{\omega}$. This weight vector might trip the search process into suboptimal regions of the non-dominated service space as the composition agent is biased towards the user preferences. In contrast, the proposed single policy multi-objective approach builds upon the composition structure to derive the relative weights among different QoS preferences. This feature allows the proposed approach to adapt efficiently to the dynamics of open Web environments where the relative weight of each QoS preference is automatically changed by the change in the dynamic environment.
task. For this end, we consider a workflow consists of 10 tasks, then we vary the number of available concrete Web services into a range of 200 to 700 per task. We run the proposed approach under these environment scales and compare the average accumulated rewards obtained with their linear weight Q-learning counterparts. As shown in Fig. 4, the proposed single policy approach outperforms the linear weight Q-learning approach despite the environment scale. The single policy approach clearly gains higher accumulated rewards throughout the learning process, and thus, leads to higher quality solutions.

In Test 2, the environment scale is represented by the number of tasks used in every workflow. For this reason, we fix the number of concrete Web services to 400 and vary the number of tasks in a range from 5 to 30. The performance of the proposed approach is also measured in terms of the average accumulated reward that the composition agent receives when the solution converges to an optimal policy. This reward value represents the aggregate $QoS$ of the optimal workflow.

Fig. 5 draws the relationship between the average accumulated rewards obtained by running the single policy multi-objective approach and the linear weight Q-learning approach multiple times with various number of tasks per workflow.

As shown in Fig. 5, the single policy multi-objective service composition approach also outperforms the linear weight Q-learning approach regardless of the number of tasks per workflow. As the number of tasks increases, the performance gap of the two approaches increases in favor of the single policy multi-objective service composition approach, which proves the scalability of the single policy approach and its ability to find high quality compositions in large-scale Web environments.

### 5.2.3 Experiment 3: Evaluation of Multiple Policy Multi-Objective Composition

The purpose of Experiment 3 is to assess the ability of the multiple policy multi-objective approach in learning the set of Pareto optimal workflows considering all the trade-offs among $QoS$ objectives. Totally four tests are carried out in this experiment. In the first three tests, each task has been assigned 50, 100 and 200 candidate Web services, respectively. Consequently, this creates a $4 \times 50$ matrix, a $4 \times 100$ and a $4 \times 200$ matrix for each service quality attribute, respectively. The multiple policy multi-objective approach is implemented and tested with the parameters given above. The proposed approach runs till the search process converges and the number of Pareto-optimal solutions/workflows are calculated accordingly.

As shown in Fig. 6, the experimental results of Test 1, Test 2 and Test 3 indicate that the proposed approach is capable of steering the search towards the non-dominated service set efficiently. As the initial attribute matrix data are created randomly, there is no idea where the true non-dominated service set is. However, it is understood that optimal workflows would be the ones with low cost, low response time, but high availability. The search process is anticipated to converge towards this direction.

The results of Test 1, as depicted in Fig. 6a, clearly show that the optimal workflows have achieved lower cost and shorter response time, but higher availability, which are centered between 0.4, 0.2, and 0.8, respectively. The results of Test 2, as depicted in Fig. 6b, support this statement, regardless of the bigger number of concrete Web services assigned to each task, as the optimal workflows continue showing the same trend with lower cost and shorter response time, but higher availability, which are centered between 0.3, 0.4, and 0.6, respectively. The results of Test 3, as depicted in Fig. 6c, also support this statement, and the optimal workflows also show the trend of lower cost and shorter response time, but higher availability, which are centered between 0.5, 0.1 and 0.8, respectively.

Test 4 is performed to display the convergence property with the consideration of different environment scales and various concrete services. Still, four tasks are considered. We experiment three different cases with the number of concrete Web services varies from 100 to 400 for each task. As shown in Fig. 7, it takes longer time to find the set of Pareto-optimal workflows 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 Pareto-optimal workflows at 800 episodes and 1000 episodes, respectively. The same tendency is anticipated to continue for any other larger number of concrete services. As a matter of fact, the three cases generated the same number of Pareto-optimal workflows, i.e., 25, at episode 400. The reason for this is currently unknown and is set for future research. In short, the proposed multiple policy multi-objective approach is able to simultaneously provide the set of Pareto-optimal workflows for service composition problems with different $QoS$ criteria.

### 5.2.4 Experiment 4: Quality Indicators

The purpose of Experiment 4 is to benchmark the results of the multiple policy approach with the state-of-the-art approaches in service composition. Towards this end, the quality of the solutions/workflows is adopted as the main comparison criterion since other criteria such as the running time or the invocation cost would heavily rely on the deployment scenario. In this experiment, we compare the quality of the solutions obtained using the multiple policy approach with one of the most well known evolutionary multi objective optimization approaches, i.e., Non
Dominated Sorting Genetic Algorithm (NSGA-II) [9]. NSGA-II has been applied widely for multi-objective service composition [16] and has shown scalable results. Yet, the application of NSGA-II requires a certain environment with known QoS parameters.

Here we use the widely used and accepted hypervolume indicator [26] as a performance metric. The hypervolume indicator measures the $n$-dimensional space contained by a solution set, i.e., the $n$-dimensional volume of the set relative to some reference point. Therefore, a solution set with a larger hypervolume presents a better set of trade-offs than a solution set with a smaller hypervolume.

A point to be considered in this comparison is the point in time at which this measurement is carried out. Many researchers have reported the results of the evaluation metric only at a single point in the process. This might be misleading as the results may change during the course of time. To avoid this, in this experiment, the results are evaluated at periodic intervals during the service composition process. For this sake, we identify the time step as the amount...
The problem of QoS-aware Web service composition is well known in SOC domain and various solutions were proposed based on different approaches [3], [10], [25]. Wang et al. [20] proposed a two-phase QoS-aware service selection approach that proved its efficiency. The first phase employs a cloud model to compute the QoS uncertainty for pruning redundant services while extracting reliable services. The second phase, then, uses Mixed Integer Programming (MIP) to select optimal services. In order to enhance the credibility of service composition plan, Lin et al. proposed a QoS-aware method [10], which took advantage of Web service’s QoS history records, rather than using the tentative QoS values advertised by the service provider. However, at last, the composition optimization problem is also instantiated into an Integer Programming
(a) Test 1: 10 tasks and 300 concrete services  
(b) Test 2: 10 tasks and 400 concrete services  
(c) Test 3: 10 tasks and 500 concrete services  
(d) Test 4: 20 tasks and 300 concrete services  
(e) Test 5: 20 tasks and 400 concrete services  
(f) Test 6: 20 tasks and 500 concrete services

Fig. 8: The hypervolume quality measurements with various scales
(IP) problem. This IP approach is hardly feasible in dynamic real-time scenarios when a large number of potential Web services are concerned [3]. Canfora et al. [5] proposed the use of Genetic Algorithms (GAs) for the QoS-aware service composition problem. The results have shown that GAs can outperform the IP used in [25], when a large number of services are available. In addition, GAs have proved to be more flexible than the MIP, since GAs allow the consideration of nonlinear composition rules. However, traditional GAs have some inherent limitations in solving QoS-aware composition problems. GAs require the pre-selection of the weights of characteristics in order to aggregate multiple objectives into a single objective function.

All the above mentioned approaches, however, cannot solve Web service selection with multiple QoS objectives and multi-constraints. They all assume that multiple quality criteria, no matter whether these criteria are competing or not, can be combined into a single criterion to be optimized, according to some utility functions. Also, when multiple quality criteria are considered, users are required to express their preferences over different, and sometimes conflicting, quality attributes as numeric weights. This is a rather demanding task since 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 [22]–[24]. Yu and Lin [24] investigated service composition with multiple QoS constraints. The composition problem is modeled as a Multi-dimension Multi-choice 0-1 Knapsack Problem (MMKP). A Multi-Constraint Optimal Path (MCOP) algorithm with heuristics is presented in [24]. However, the parameters aggregation using the Min function is neglected. The Web Service Agent Framework (WSAF) has been proposed by Maximilien and Singh [12] in order to achieve service selection by considering the preferences of several service consumers, as well as, the trustworthiness of providers.

The application of Evolutionary Algorithms (EAs) to the multi-objective Web service composition has been witnessing a growing interest [16], [17]. Claro et al. [8] discussed the advantages of Multi-Objective Genetic Algorithms (MOGA) in Web service selection and a popular multi-objective algorithm, NSGA-II [9], is used to find optimal sets of Web services. Wada et al. [16] proposed an optimization framework based on NSGA-II, as well. They claimed lower time complexity than the NSGA-II, however, they pointed out that the solution quality might fluctuate due to the randomness of the search approach. Other EAs that have been proposed to solve multi-objective service composition include, Multi-Objective Particle Swarm Optimizer (MOPSO) [6], and Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [14]. These EAs based approaches employ mathematical improvements to solve multi-objective service composition problems. However, as the dimensionality of the problem increases, the performance of these EAs based approaches significantly deteriorates, since they cannot find a wide range of alternative solutions. In addition, MOGA and MOPSO cannot solve optimization problems with concave Pareto fronts which are commonly encountered in the real Web environments. In contrast, the MORL based approaches, proposed in this paper, are able to explore well the Pareto front of multi-objective service composition problems and deliver optimal solutions.

In addition, EAs based approaches require a level of awareness, i.e., a clear model, of the service composition domain to setup the initial population through encoding the available service combinations as genomes. In contrast, our MORL based approaches can learn how to best select Web services in complex Web environments based on multiple QoS criteria without any prior knowledge regarding the nature or the dynamics of these Web environments. Up to our knowledge, our approaches are the first attempt to use MORL to solve the multi-objective service composition problem.

An alternative branch of work aims at calculating the real Pareto front, explicitly, using the skyline operator [1], [22], [23]. Yu and Bouguettaya [22], [23] presented three algorithms for calculating all the Pareto optimal services in a bottom up fashion. The first, the One-Pass Algorithm, enumerates the available services, and prunes the dominated ones. The second, the Dual Progressive Algorithm, reports the Pareto optimal services incrementally. The third, the Bottom Up Algorithm, improves the efficiency by calculating the Pareto set for larger parts of the workflow. However, such algorithms cannot have polynomial time complexity, because the size of the Pareto front may grow exponentially, in the number of workflow tasks. In addition, the usage of the skyline operator to prune the search space may exclude some potential services before the selection phase.

7 DISCUSSIONS

Each of the proposed MORL based approaches, in this paper, exhibits a set of advantages that makes it suitable to a certain class of service composition problems. The multiple policy multi-objective service composition approach produces a set of optimal solutions which enables a posteriori decision about the solution to be selected. In addition, the presentation of these solutions gives the user better knowledge of the trade-offs available, as well as, better insights into the interaction amongst the competing objectives. However, the primary disadvantage of gathering multiple solutions is the increased computational cost and time spent on the interaction with the environment.
Therefore, this approach is recommended to small to medium size multi-objective service composition problems. On the other hand, while the single policy multi-objective service composition approach introduces only one optimal solution, and therefore denies the user the chance of the posteriori decision, it can reduce the computational cost and time relative to searching for a set of optimal solutions, which is extremely important in open and dynamic Web environments. For this reason, this approach is recommended to large-scale service composition problems. It is worth noting that the computational cost is represented by how often (the number of episodes) the agent ought to interact with the service environment. This is considered a robust measurement as the sole values of interaction time or service fee might be sensitive to the network conditions. Also, while the online learning scheme adopted by the proposed approaches fits dynamic composition scenarios, other approaches, e.g., combining learning and optimization, may be better for other cases.

8 CONCLUSION

This paper proposes two novel approaches to resolve the QoS-aware multi-objective service composition problems. The first approach addresses the single policy multi-objective composition scenarios, while the second approach addresses the multiple policy multi-objective composition scenarios. The simulation results have shown the abilities of the proposed approaches to efficiently compose Web services based on multiple QoS criteria, especially in uncertain and dynamic environments. The future work is set to study the impact of the relationship among multiple quality objectives.

REFERENCES


Ahmed Moustafa received the BSc and the MSc degrees from Zagazig University, Egypt in 2003 and 2010, respectively. Now, he is working toward the PhD degree in computer science at the University of Wollongong, Australia. His research interests focus on self-organizing multi-agent systems and their applications.

Minjie Zhang received the BSc and the MSc degrees from Fudan University, P.R. China in 1982 and the PhD degree in computer science from the University of New England, Australia, in 1996. Now, she is an associate professor of computer science at the University of Wollongong, Australia. Her research interests include multiagent systems and agent-based modeling in complex domains. She is a senior member of IEEE.